Scientific Computing and Programming Problems

by
Willi-Hans Steeb
International School for Scientific Computing
at
University of Johannesburg, South Africa

Yorick Hardy
Department of Mathematical Sciences
at
University of South Africa, South Africa

George Dori Anescu
email: george.anescu@gmail.com

Preface

The purpose of this book is to supply a collection of problems in matrix calculus.

Prescribed books for problems.

1) Matrix Calculus and Kronecker Product with Applications and C++ Programs

by Willi-Hans Steeb
World Scientific Publishing, Singapore 1997
ISBN 981 023 2411
http://www.worldscibooks.com/mathematics/3572.html

2) Problems and Solutions in Introductory and Advanced Matrix Calculus

by Willi-Hans Steeb
World Scientific Publishing, Singapore 2006
ISBN 981 256 916 2
http://www.worldscibooks.com/mathematics/6202.html

3) Continous Symmetries, Lie Algebras, Differential Equations and Computer Algebra, second edition

by Willi-Hans Steeb
World Scientific Publishing, Singapore 2007
ISBN 981-256-916-2
http://www.worldscibooks.com/physics/6515.html

4) Problems and Solutions in Quantum Computing and Quantum Information, second edition

by Willi-Hans Steeb and Yorick Hardy
World Scientific, Singapore, 2006
ISBN 981-256-916-2
http://www.worldscibooks.com/physics/6077.html

The International School for Scientific Computing (ISSC) provides certificate courses for this subject. Please contact the author if you want to do this course or other courses of the ISSC.

e-mail addresses of the author:

steebwilli@gmail.com
steeb_wh@yahoo.com

Home page of the author:

http://issc.uj.ac.za

Contents

Notation

$:=$	is defined as
$\in$	belongs to (a set)
$\notin$	does not belong to (a set)
$\cap$	intersection of sets
$\cup$	union of sets
$\emptyset$	empty set
$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integers
$\mathbb{Q}$	set of rational numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}^+$	set of nonnegative real numbers
$\mathbb{C}$	set of complex numbers
$\mathbb{R}^n$	n-dimensional Euclidean space
	space of column vectors with n real components
$\mathbb{C}^n$	n-dimensional complex linear space
	space of column vectors with n complex components
$\mathcal{H}$	Hilbert space
i	$\sqrt{-1}$
$\Re z$	real part of the complex number z
$\Im z$	imaginary part of the complex number z
$\|z\|$	modulus of complex number z
	$\|x+iy\| = (x^2+y^2)^{1/2}, \;\; x, y \in \mathbb{R}$
$T \subset S$	subset T of set S
$S \cap T$	the intersection of the sets S and T
$S \cup T$	the union of the sets S and T
$f(S)$	image of set S under mapping f
$f \circ g$	composition of two mappings $(f \circ g)(x) = f(g(x))$
$\mathbf{x}$	column vector in $\mathbb{C}^n$
$\mathbf{x}^T$	transpose of $\mathbf{x}$ (row vector)
$\mathbf{0}$	zero (column) vector
$\|\,.\,\|$	norm
$\mathbf{x} \cdot \mathbf{y} \equiv \mathbf{x}^* \mathbf{y}$	scalar product (inner product) in $\mathbb{C}^n$
$\mathbf{x} \times \mathbf{y}$	vector product in $\mathbb{R}^3$
A, B, C	$m \times n$ matrices
$\det(A)$	determinant of a square matrix A
tr(A)	trace of a square matrix A
$\mathrm{rank}(A)$	rank of matrix A
A^T	transpose of matrix A
$\overline{A}$	conjugate of matrix A

A^*	conjugate transpose of matrix A
$A^\dagger$	conjugate transpose of matrix A (notation used in physics)
A^{-1}	inverse of square matrix A (if it exists)
I_n	$n \times n$ unit matrix
I	unit operator
0_n	$n \times n$ zero matrix
AB	matrix product of $m \times n$ matrix A and $n \times p$ matrix B
$A \bullet B$	Hadamard product (entry-wise product) of $m \times n$ matrices A and B
$[A, B] := AB - BA$	commutator for square matrices A and B
$[A, B]_+ := AB + BA$	anticommutator for square matrices A and B
$A \otimes B$	Kronecker product of matrices A and B
$A \oplus B$	Direct sum of matrices A and B
δ_{jk}	Kronecker delta with $\delta_{jk} = 1$ for $j = k$ and $\delta_{jk} = 0$ for $j \neq k$
λ	eigenvalue
ϵ	real parameter
t	time variable
$\hat{H}$	Hamilton operator

The Pauli spin matrices are used extensively in the book. They are given by

$$\sigma_1 := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

In some cases we will also use σ_x, σ_y and σ_z to denote σ_1, σ_2 and σ_3.

Chapter 1

Quickies

Problem 1. Can the expression $\sqrt{3-2\sqrt{2}}$ be simplified for computation? Hint. Let $a>0$ and $b>0$. Calculate $(\sqrt{a}-b)(\sqrt{a}-b)$ and compare coefficients.

Solution 1. We have

$$(\sqrt{a}-b)(\sqrt{a}-b)=a+b^2-2b\sqrt{a}.$$

Thus we find $a=2$ and $b=1$ and the simplified expression is $\sqrt{2}-1$.

Problem 2. Let n be a positive integer and $x,y\in\mathbb{R}$. Can the calculation of

$$\sum_{k=0}^{n}\binom{n}{k}x^{n-k}y^k$$

be simplified for computation?

Solution 2. We have the identity (Binomial theorem)

$$\sum_{k=0}^{n}\binom{n}{k}x^{n-k}y^k\equiv(x+y)^n.$$

Problem 3. Let $n\in\mathbb{N}$. How can the calculation of

$$x(e^{-x}+e^{-2x}+\cdots+e^{-nx})$$

be simplified for n large?

Solution 3. We can use the identity

$$x(e^{-x} + e^{-2x} + \cdots + e^{-nx}) \equiv \frac{x}{e^x - 1}(1 - e^{-nx}).$$

Problem 4. Let $\mathbb{Z}$ be the integer numbers. Let $\mathbb{N}_0$ be the natural numbers including 0. Find a $1-1$ map

$$f : \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}_0$$

with $f(0,0,0) = 0$ and $f(1,0,0) = 1$. The number of nearest neighbours are 6. Let $(j_1, j_2, j_3) \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$. Then the six nearest neighbours are

$$(j_1 + 1, j_2, j_3), \quad (j_1, j_2 + 1, j_3), \quad (j_1, j_2, j_3 + 1)$$

$$(j_1 - 1, j_2, j_3), \quad (j_1, j_2 - 1, j_3), \quad (j_1, j_2, j_3 - 1).$$

Give a C++ implementation using the `Verylong` class of SymbolicC++. Give a Java implementation using the `BigInteger` class.

Solution 4.

Problem 5. (i) Let $\epsilon \in \mathbb{R}$ and $\epsilon > 0$. Let $\mathbf{v}$ be a nonzero vector in $\mathbb{R}^n$. Assume that $\|\mathbf{v}\| \gg \epsilon$. Show that

$$\sqrt{\mathbf{v}^T\mathbf{v} \pm \epsilon} \approx \|\mathbf{v}\| \pm \frac{\epsilon}{2\mathbf{v}^T\mathbf{v}}.$$

(ii) Let $x, \ell \geq 0$ and $x \ll \ell$. Show that

$$\sqrt{\ell^2 + x^2} - \ell \approx \frac{x^2}{2\ell}.$$

Solution 5.

Problem 6. (i) Let N_1, N_2 be given positive integers. Let $n_1 = 0, 1, \ldots, N_1 - 1$, $n_2 = 0, 1, \ldots, N_2 - 1$. There are $N_1 \cdot N_2$ points. The points (n_1, n_2) are a subset of $\mathbb{N}_0 \times \mathbb{N}_0$ and can be mapped one-to-one onto a subset of $\mathbb{N}_0$

$$j(n_1, n_2) = n_1 N_2 + n_2$$

where $j = 0, 1, \ldots, N_1 \cdot N_2 - 1$. Find the inverse of this map. Consider first the case $N_1 = N_2 = 2$.
(ii) Give a C++ implementation of the map and the inverse.

Solution 6. (S) (i) Let $x \in \mathbb{R}$. Let $\lfloor x \rfloor$ denote the integer which is not greater than x. For $N_1 = N_2 = 2$ we have the map

$$(0,0) \leftrightarrow 0, \quad (0,1) \leftrightarrow 1, \quad (1,0) \leftrightarrow 2, \quad (1,1) \leftrightarrow 3.$$

The inverse map is given by

$$n_1 = \left\lfloor \frac{j}{2} \right\rfloor, \qquad n_2 = j - 2 \left\lfloor \frac{j}{2} \right\rfloor.$$

For general N_1 and N_2 we have

$$n_1 = \left\lfloor \frac{j}{N_2} \right\rfloor, \qquad n_2 = j - N_2 \left\lfloor \frac{j}{N_2} \right\rfloor.$$

(ii) Utilizing integer division in C++ the implementation is

```
// OneTwoInverse.cpp

#include <iostream>
using namespace std;

int main(void)
{
  int N1, N2; N1 = 8; N2 = 5;
  int n1, n2; n1 = 0; n2 = 0;
  int j = 0;
  for(n1=0;n1 < N1;n1++)
   for(n2=0;n2 < N2;n2++)
   {
   j = n1*N2 + n2;
   cout << j << "(" << n1 << "," << n2 << ")" << endl;
   }
   cout << endl;
   // inverse
   int np = N1*N2;
   for(j=0;j < np;j++)
   {
   n1 = j/N2;          // j/N2 is integer division
   n2 = j-N2*(j/N2); // j/N2 is integer division
   cout << j << "(" << n1 << "," << n2 << ")" << endl;
   }
   return 0;
}
```

Problem 7. Let $n \in \mathbb{N}$. The map $f : [0, 2n] \to [0, 2n]$ on the integers defined by

$$f(0) = n$$

$$f(k) = 2n + 1 - k \text{ for } 0 < k \leq n$$
$$f(k) = 2n - k \text{ for } n < k \leq 2n$$

plays a role for the converse of Sarkovskii's theorem.
(i) Let $n = 2$. Starting with 1 find

$$f(1),\ f(f(1)),\ f(f(f(1))),\ f(f(f(f(1)))),\ f(f(f(f(f(1))))).$$

Discuss.
(ii) Give a C++ implementation of this map. The user provides the n.

Solution 7. (i) We have

$$f(1) = 4, \quad f(4) = 0, \quad f(0) = 2, \quad f(2) = 3, \quad f(3) = 1.$$

Thus we have periodic orbit.

=

Problem 8. Show that

$$\frac{1}{a+n-k+1}\left(\frac{1}{a-b+1} + \frac{1}{n-k+b}\right) = \frac{1}{(a-b+1)(n-k+b)}.$$

Solution 8.

Problem 9. Given a vector of length n. Write a C++ program that checks whether all entries are pairwise different.

Solution 9.

```
// pairwisedifferent.cpp

#include <iostream>
using namespace std;

bool pairwise(double* v,int n)
{
 for(int i=0;i<n;i++)
  for(int j=i+1;j<n;j++) if(v[i]==v[j]) return false;
  return true;
}

int main(void)
{
```

```
  int n = 6;
  double* v; v = new double[n];
  v[0] = 1.2; v[1] = 1.7; v[2] = 2.1;
  v[3] = 1.1; v[4] = 1.5; v[5] = 1.9;
  bool b = pairwise(v,n);
  cout << "b = " << b << endl;
  delete[] v;
  return 0;
}
```

Problem 10. The *sinc function* $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = \frac{\sin(\pi x)}{\pi x}$$

can be evaluated using the series expansion

$$f(x) = 1 - \frac{1}{3!}(\pi x)^2 + \frac{1}{5!}(\pi x)^4 - \cdots$$

However the sinc function could also be evaluated from

$$f(x) = \prod_{k=1}^{\infty} \left(1 - \frac{x^2}{k^2}\right).$$

Compare the two methods.

Solution 10.

Problem 11. The quadratic equation $x^2 = x + 1$ has the solutions

$$\tau = \frac{1}{2}(1 + \sqrt{5}), \qquad \sigma = \frac{1}{2}(1 - \sqrt{5})$$

(golden mean numbers). Let $k \in \mathbb{Z}$. Can the expressions

$$\tau^{k-1} + \tau^{k-2}, \qquad \sigma^{k-1} + \sigma^{k-2}$$

be simplified?

Solution 11. We have the identities

$$\tau^k = \tau^{k-1} + \tau^{k-2}, \qquad \sigma^k = \sigma^{k-1} + \sigma^{k-2}.$$

Problem 12. How would one calculate more efficiently (i.e. minimizing the number of multiplications) the analytic function $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = x + 2x^2 + 3x^3$$

for a given x.

Solution 12. We have

$$f(x) = x(1 + 2x + 3x^2) = x(1 + x(2 + 3x))$$

i.e. we have 3 multiplications and 2 additions, whereas for $x+2x^2+3x^3$ we have five multiplications and two additions.

Problem 13. Consider the analytic function $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = \frac{x}{1 + x^2}.$$

Simplify the calculation of the integral

$$\int_{-1}^{2} f(x)dx.$$

Hint. First show that $f(x) = -f(-x)$.

Solution 13. Since $f(x) = -f(-x)$ we have

$$\int_{-1}^{2} f(x)dx = \int_{-1}^{1} f(x)dx + \int_{1}^{2} f(x)dx = \int_{1}^{2} f(x)dx.$$

Problem 14. (S) Solve the quadratic equation $\omega^2 + \omega + 1 = 0$ by multiplying this equation with ω and inserting the quadratic equation and then solving the resulting cubic equation. Select the solutions from the cubic equation which are also solutions of the quadratic equation. Note that $1 \equiv \exp(i2\pi)$.

Solution 14. Multiplying the quadratic equation with ω and inserting then the quadratic equation yields $\omega^3 = 1$. This equation has the three solutions $\omega_1 = 1$, $\omega_2 = e^{i2\pi/3}$, $\omega_3 = e^{i4\pi/3}$. Obviously ω_2 and ω_3 are the solution to the quadratic equation, whereas ω_1 is not a solution of the quadratic equation.

Problem 15. Find numerically solutions of the transcendental equation

$$e^{-x} + \frac{x}{5} - 1 = 0$$

for $x \geq 0$.

Solution 15. Let $f(x) = e^{-x} + x/5 - 1$. For $x = 0$ we have $f(0) = 1/5 > 0$. For $x = 5$ we have $e^{-5} > 0$. $x \approx 4.96$.

Problem 16. Let $n_0, n_1, n_2 \in \mathbb{N}_0$. Implement the function

$$f(n_0, n_1, n_2) = \frac{(n_0 + n_1 + n_2)!}{n_0! n_1! n_2!}$$

in SymbolicC++ utilizing the `Verylong` class and `Rational` class.

Solution 16.

Problem 17. Calculate efficiently

$$\int_0^7 \sin(x)dx, \qquad \int_0^7 \cos(x)dx.$$

Solution 17.

$$=$$

Problem 18. (S) Let $i, j, k \in \mathbb{N}_0$. Find all solutions of $i + j + k = 3$. Give the solution in lexicographical order.

Solution 18. There are 10 solutions given in lexicographical order

$$003, 012, 021, 030, 102, 111, 120, 201, 210, 300.$$

Problem 19. The number π can be calculated from the expansion

$$\pi = \sum_{j=0}^{\infty} \frac{(j!)^2 2^{j+1}}{(2j+1)!}.$$

Let n be positive integer. Give an implementation of the sum

$$s = \sum_{j=0}^{n} \frac{(j!)^2 2^{j+1}}{(2j+1)!}$$

with SymbolicC++ using the `Verylong` and `Rational` class and so find an approximation of π.

Solution 19.

Problem 20. Let $\mathbb{N}_0$ be the set of natural numbers including 0. Let $n_1, n_2, n_3 \in \mathbb{N}_0$. An invertible function $f : \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0 \mapsto \mathbb{N}_0$ is defined as follows. Let

$n = n_1 + n_2 + n_3$. For a fixed n we have $f(n) < f(n+1)$ and within a fixed n a lexicographical ordering is assumed. For the first ten elements one has

$$(0,0,0) \to 0,\ (0,0,1) \to 1,\ (0,1,0) \to 2,\ (1,0,0) \to 3,$$

$$(0,0,2) \to 4,\ (0,1,1) \to 5,\ (1,0,1) \to 7,\ (1,1,0) \to 8,\ (2,0,0) \to 9$$

Give a C++ implementation of the function f and its inverse f^{-1} using templates so that the `Verylong` class of SymbolicC++ can be used. Give a Java implementation using the `BigInteger` class.

Solution 20.

Problem 21. Let $f \in L_2(\mathbb{R})$. *Poisson's summation formula* in one dimension is given by

$$\sum_{n=-\infty}^{+\infty} f(n) = \sum_{q=-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x) e^{-2\pi i q x} dx.$$

Show that if f is an even function of x, then the summation formula can be written as

$$\sum_{n=1}^{+\infty} f(n) = -\frac{1}{2} f(0) + \int_0^{+\infty} f(x) dx + 2 \sum_{q=1}^{+\infty} \int_0^{+\infty} f(x) \cos(2\pi q x) dx.$$

Apply it to the function $f(x) = e^{-|x|}$.

Solution 21.

Problem 22. The *Catalan constant* is defined as

$$G = 1 - 3^{-2} + 5^{-2} - 7^{-2} + \cdots$$

Give a SymbolicC++ implementation using the `Verylong` and `Rational` class to find an approximation of the constant.

Solution 22.

Problem 23. Let $n \in \mathbb{Z}$. Simplify $\sin(n\pi)$, $\cos(2n\pi)$, $\cos((2n+1)\pi)$.

Solution 23. We have $\sin(n\pi) = 0$, $\cos(2n\pi) = 1$, $\cos((2n+1)\pi) = -1$.

Problem 24. Consider the normalized vectors

$$\mathbf{n}_j := \begin{pmatrix} \sin(\theta_j)\cos(\phi_j) \\ \sin(\theta_j)\sin(\phi_j) \\ \cos(\theta_j) \end{pmatrix}, \quad \mathbf{n}_k := \begin{pmatrix} \sin(\theta_k)\cos(\phi_k) \\ \sin(\theta_k)\sin(\phi_k) \\ \cos(\theta_k) \end{pmatrix}$$

in $\mathbb{R}^3$. Find the scalar product

$$\mathbf{n}_j \cdot \mathbf{n}_k$$

and simplify it. The scalar product is the angle between the two vectors.

Solution 24. We obtain (check)

$$\mathbf{n}_j \cdot \mathbf{n}_k = \cos(\theta_j)\cos(\theta_k) + \sin(\theta_j)\sin(\theta_k)\cos(\phi_j - \phi_k).$$

Problem 25. (i) Let $\mathbf{u}$, $\mathbf{v}$ be (column) vectors in the Euclidean space $\mathbb{R}^n$. Now $\mathbf{u}^T$, $\mathbf{v}^T$ are the corresponding row vectors (^{T} denotes transpose) and thus $\mathbf{u}^T\mathbf{v}$ is the scalar product of $\mathbf{u}$ and $\mathbf{v}$. What does

$$A := \sqrt{|(\mathbf{u}^T\mathbf{u})(\mathbf{v}^T\mathbf{v}) - (\mathbf{u}^T\mathbf{v})^2|}$$

calculate?
(ii) Consider $\mathbb{R}^4$ and the vectors

$$\mathbf{u} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \mathbf{v} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Calculate A.

Solution 25. (i) The quantity A is the area spanned by the vectors $\mathbf{u}$ and $\mathbf{v}$.
(ii) For the given example we find $A = 2$.

Problem 26. (S) (i) Let $\mathbf{u}$, $\mathbf{v}$ be column vectors in $\mathbb{C}^n$ and thus $\mathbf{u}^*$, $\mathbf{v}^*$ (transpose and complex conjugate) are row vectors. Calculate efficiently

$$\mathrm{tr}(\mathbf{u}\mathbf{u}^*\mathbf{v}\mathbf{v}^*).$$

Note that $\mathbf{u}\mathbf{u}^*\mathbf{v}\mathbf{v}^*$ is an $n \times n$ matrix. Could one utilize that matrix multiplication is associative? Discuss. Is

$$\mathrm{tr}(\mathbf{u}\mathbf{u}^*\mathbf{v}\mathbf{v}^*) \geq 0\,?$$

Prove or disprove.
(ii) Let A be a 2×2 matrix. Calculate efficiently $\mathrm{tr}(A^2)$.

Solution 26.

Problem 27. Consider the 2×2 matrices

$$C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad A = \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{11} \end{pmatrix}, \qquad a_{11}, a_{12} \in \mathbb{R}.$$

Can the expression

$$A^3 + 3AC(A + C) + C^3$$

be simplified for computation?

Solution 27. Since $AC = CA$ we can write the expression as $(A + C)^3$.

Problem 28. Given two invertible $n \times n$ matrices A and B.
(i) How can we calculate $B^{-1}A^{-1}$ more efficiently?
(ii) Let $\otimes$ be the Kronecker product. How can we calculate $B^{-1} \otimes A^{-1}$ more efficiently?

Solution 28. (i) We can utilize the identity

$$(AB)^{-1} \equiv B^{-1}A^{-1}.$$

(ii) We can utilize the identity

$$(B \otimes A)^{-1} \equiv B^{-1} \otimes A^{-1}.$$

Problem 29. (i) Let

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \in \mathbb{R}^3.$$

What does

$$\frac{1}{2} \det \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}$$

calculate?
(ii) Consider the coordinates

$$\mathbf{p}_1 = (x_1, y_1, z_1)^T, \quad \mathbf{p}_2 = (x_2, y_2, z_2)^T, \quad \mathbf{p}_3 = (x_3, y_3, z_3)^T$$

with $\mathbf{p}_1 \neq \mathbf{p}_2$, $\mathbf{p}_2 \neq \mathbf{p}_3$, $\mathbf{p}_3 \neq \mathbf{p}_1$. We form the vectors

$$\mathbf{v}_{21} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}, \quad \mathbf{v}_{31} = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix}.$$

Let $\times$ be the vector product. What does

$$\frac{1}{2}|\mathbf{v}_{21} \times \mathbf{v}_{31}|$$

calculate? Apply it to $\mathbf{p}_1 = (0, 0, 0)^T$, $\mathbf{p}_2 = (1, 0, 1)^T$, $\mathbf{p}_3 = (1, 1, 1)^T$.

Solution 29.

Problem 30. What is the output of the following C++ program

```
// whileloop.cpp

#include <iostream>
using namespace std;

int main(void)
{
  int x = 0; int t = 0; int p = 0;
  while(t < 100) {
   if(p==0) x = x + 2;
   if(p==1) x = x - 1;
  p = 1 - p;
  t++;
  } // end while
  cout << "x = " << x << endl;
  cout << "p = " << p << endl;
  cout << "t = " << t << endl;
  return 0;
}
```

Solution 30.

Problem 31. The surface area of a torus with inner radius a and outer radius b is a given by

$$A = \pi^2(b^2 - a^2).$$

The formula for the volume of a torus is given by

$$V = \frac{\pi^2}{4}(a+b)(b-a)^2.$$

Simplify the calculation of V given A.

Solution 31. Since (check)

$$V = \frac{\pi^2}{4}(a+b)(b-a)^2 \equiv \frac{\pi^2}{4}(b^2-a^2)(b-a).$$

Thus

$$V = \frac{1}{4}A(b-a).$$

Problem 32. Let a, b be non-negative integers.

(i) Simplify the expression

$$E_1 = \sqrt{a + \sqrt{-b}} + \sqrt{a - \sqrt{-b}}.$$

(ii) Simplify the expression

$$E_2 = \sqrt{a + \sqrt{-b}} - \sqrt{a - \sqrt{-b}}.$$

Solution 32. (i) We set

$$c := \frac{1}{2}(\sqrt{a^2 + b} + a).$$

Then E_1 can be written as $E_1 = 2\sqrt{c}$, i.e. E_1 is a real number.
(ii) We set

$$d := \frac{1}{2}(\sqrt{a^2 + b} - a).$$

Then E_2 can be written as $E_1 = 2\sqrt{-d}$.

Problem 33. Let $x \in [-1, 1]$. Simplify $\arcsin(x) + \arccos(x)$.

Solution 33. One has

$$\arcsin(x) + \arccos(x) = \frac{\pi}{2}.$$

Problem 34. Simplify

$$f(\alpha, \beta, \gamma) = \sin(\alpha + \beta - \gamma) + \sin(\beta + \gamma - \alpha) + \sin(\gamma + \alpha - \beta) - \sin(\alpha + \beta + \gamma).$$

Solution 34. We have

$$f(\alpha, \beta, \gamma) = 4 \sin(\alpha) \sin(\beta) \sin(\gamma).$$

Problem 35. Let $f : \mathbb{R} \to \mathbb{R}$ be an analytic function. Consider the central *difference operator* δ defined by

$$\delta(f(x)) := f(x + \frac{1}{2}h) - f(x - \frac{1}{2}h)$$

where $h > 0$ is the step length. The operator δ is linear. Find $\delta(\delta(f(x))$.

Solution 35. We have

$$\begin{aligned}\delta(\delta(f(x)) &= \delta(f(x+\frac{1}{2}h) - f(x-\frac{1}{2}h)) \\ &= \delta(f(x+\frac{1}{2}h)) - \delta(f(x-\frac{1}{2}h)) \\ &= f(x+h) - f(x) - f(x) + f(x-h) \\ &= f(x+h) - 2f(x) + f(x-h).\end{aligned}$$

Problem 36. Consider the coordinates in $\mathbb{R}^3$

$$\mathbf{p}_1 = (x_1, y_1, z_1), \quad \mathbf{p}_2 = (x_2, y_2, z_2), \quad \mathbf{p}_3 = (x_3, y_3, z_3)$$

with $\mathbf{p}_1 \neq \mathbf{p}_2$, $\mathbf{p}_2 \neq \mathbf{p}_3$, $\mathbf{p}_3 \neq \mathbf{p}_1$. We form the two vectors

$$\mathbf{v} = \mathbf{p}_2 - \mathbf{p}_1, \qquad \mathbf{w} = \mathbf{p}_3 - \mathbf{p}_1.$$

What does $\frac{1}{2}|\mathbf{v} \times \mathbf{w}|$ calculate?

Solution 36.

Problem 37. Let n be a positive integer. Give a C++ implementation of sum

$$\sum_{k=0}^{n} \sum_{\ell=0}^{n} \binom{k}{\ell}$$

using templates so that the `Verylong` class of SymbolicC++ can be used.

Solution 37.

Problem 38. Let $x \in \mathbb{R}$. Show that

$$\frac{1}{e^{-x}+1} \equiv 1 - \frac{1}{e^x+1}.$$

Solution 38.

Problem 39. Find a good approximation of $\sqrt{29}$ utilizing

$$29 \equiv 36\left(1 - \frac{7}{36}\right) \equiv 6^2\left(1 - \frac{7}{36}\right)$$

and an expansion.

Solution 39. SCandPP

Problem 40. Let $n \in \mathbb{Z}$. Show that

$$\cos((n+1)\alpha) + \cos((n-1)\alpha) \equiv 2\cos(\alpha)\cos(n\alpha)$$
$$\sin((n+1)\alpha) + \sin((n-1)\alpha) \equiv 2\cos(\alpha)\sin(n\alpha).$$

Solution 40.

Problem 41. Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function. Assume that $f(x) = -f(-x)$. Let $a < 0$ and $b > 0$. Simplify the calculation of

$$\int_a^b f(x)dx.$$

Solution 41.

Problem 42. Calulate approximative $(72)^{1/6}$ utilizing the expression

$$(72)^{1/6} = 2\left(1 + \frac{1}{8}\right)^{1/6}.$$

Problem 43. (i) Calculate $\sqrt{2}$ utilizing

$$\sqrt{2} = \frac{7}{5}\left(1 - \frac{1}{50}\right)^{-1/2}.$$

(ii) Calculate $\frac{1}{2}(1+\sqrt{5})$ utilizing

$$\frac{1}{2}\left(1 + \sqrt{5}\right) = \frac{1}{2} + \left(1 - \frac{1}{5}\right)^{-1/2}.$$

(iii) Calculate $\ln(2)$ utilizing the hypergeometric representation of $x^{-1}\ln(1-x)$ for $x = 1/2$.

Solution 43.

Problem 44. Show that

$$\frac{1}{e^x - 1} \equiv -\frac{1}{e^{-x} - 1} - 1.$$

Solution 44. We have

$$\frac{1}{e^x - 1} = \frac{1}{e^x - 1}\frac{e^{-x}}{e^{-x}} = \frac{e^{-x}}{1 - e^{-x}}$$

$$= -\frac{e^{-x}}{e^{-x}-1}$$
$$= -\frac{1}{e^{-x}-1} - 1.$$

Problem 45. Let $\mathbf{q}, \mathbf{q}_1, \mathbf{q}_2, \mathbf{x} \in \mathbb{R}^n$ and $\epsilon \in \mathbb{R}$. Simplify the expression

$$\|\mathbf{q} - \frac{\epsilon}{2}\mathbf{x} - \mathbf{q}_1\|^2 + \|\mathbf{q} + \frac{\epsilon}{2}\mathbf{x} - \mathbf{q}_2\|^2$$

where $\|\,.\,\|$ denotes the Euclidean norm in $\mathbb{R}^n$. Apply the scalar product in $\mathbb{R}^n$.

Solution 45. We obtain

$$\|\mathbf{q}-\frac{\epsilon}{2}\mathbf{x}-\mathbf{q}_1\|^2+\|\mathbf{q}+\frac{\epsilon}{2}\mathbf{x}-\mathbf{q}_2\|^2 = \|\mathbf{q}-\mathbf{q}_1\|^2+\|\mathbf{q}-\mathbf{q}_2\|^2+\frac{\epsilon^2}{2}\|\mathbf{x}\|^2+\epsilon\mathbf{x}\cdot(\mathbf{q}_1-\mathbf{q}_2).$$

Problem 46. Let $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ be vectors in $\mathbb{R}^3$. Show that

$$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) \equiv (\mathbf{u} \cdot \mathbf{w})\mathbf{v} - (\mathbf{u} \cdot \mathbf{v})\mathbf{w}.$$

Solution 46.

Problem 47. Show that

$$\frac{1}{2}(1 + \cos(\theta)) \equiv \cos^2(\theta/2), \qquad \frac{1}{2}(1 - \cos(\theta)) \equiv \sin^2(\theta/2).$$

Solution 47.

Chapter 2

Bitwise Operations

Problem 1. A boolean function $f : \{0,1\}^n \to \{0,1\}$ ($x_j \in \{0,1\}, j = 1, \ldots, n$) can be transformed from the domain $\{0,1\}$ into the spectral domain by a linear transformation

$$T\mathbf{y} = \mathbf{s}$$

where T is a $2^n \times 2^n$ orthogonal matrix, $\mathbf{y} = (y_0, y_1, \ldots, y_{2^n-1})^T$ is the two-valued ($\{+1, -1\}$ with $0 \leftrightarrow 1$, $1 \leftrightarrow -1$) truth table of the boolean function and spectral coefficients s_j. Since T is invertible we have

$$T^{-1}\mathbf{s} = \mathbf{y}.$$

For T we select the Hadamard matrix. The $(n+1) \times (n+1)$ Hadamad matrix is recursively defined as

$$H(n) = \begin{pmatrix} H(n-1) & H(n-1) \\ H(n-1) & -H(n-1) \end{pmatrix}, \quad n = 1, 2, \ldots$$

with $H(0) = (1)$ ((1×1) matrix). The inverse of $H(n)$ is given by

$$H(n)^{-1} = \frac{1}{2^n} H(n).$$

Now any boolean function can be expanded as the arithmetical polynomial

$$f(x_1, \ldots, x_n) = \frac{1}{2^{n+1}}(2^n - s_0 - s_1(-1)^{x_n} - s_2(-1)^{x_{n-1}} - \cdots - s_{2^n-1}(-1)^{x_1 \oplus x_2 \oplus \cdots \oplus x_n}$$

where $\oplus$ denotes the modulo-2 addition and the

$$(s_0, s_1, \ldots, s_{2^n-1}) = \mathbf{s}$$

are the spectral coefficients. Consider the boolean function $f : \{0,1\}^3 \to \{0,1\}$

$$f(x_1, x_2, x_3) = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 + x_1 \cdot x_2 \cdot \bar{x}_3.$$

Find the truth table, the vector $\mathbf{y}$ and then using $H(3)$ calculate the spectral coefficients s_j $(j = 0, 1, \ldots, 7)$.

Solution 1. The truth table is given by

```
x_1 x_2 x_3 f(x_1,x_2,x_3) y
0   0   0
0   0   1
0   1   0
0   1   1
1   0   0
1   0   1
1   1   0
1   1   1
```

where we used the map $0 \to 1$ and $1 \to -1$ to find the vector $\mathbf{y}$. Now the Hadamard matrix $H(3)$ is the 8×8 matrix (calculate)

$$H(3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Problem 2. Analogously to the Hamming distance for finite sequences, a metric can be used to compute distances between infinite $u(j)$ and $v(j)$, where $j \in \mathbb{Z}$

$$d(u, v) := \sum_{j \in \mathbb{Z}} \frac{|u(j) - v(j)|}{2^{|j|}}.$$

Consider the infinite alternating sequences

```
...010101010...  = u
...101010101...  = v
       ^
       |
       0
```

Find the distance between u and v.

Solution 2.

Problem 3. Let $x, y \in \{0,1\}$ and $\cdot$ the AND operation. Is the circuit

$$x' = x, \qquad y' = x \cdot y$$

a reversible gate?

Solution 3. The truth table is
Thus is gate is not reversible.

Problem 4. Consider the truth table

$$\begin{aligned} &(0,0,0) \mapsto 1, \quad (0,0,1) \mapsto 0 \\ &(0,1,0) \mapsto 0, \quad (0,1,1) \mapsto 0 \\ &(1,0,0) \mapsto 0, \quad (1,0,1) \mapsto 0 \\ &(1,1,0) \mapsto 0, \quad (1,1,1) \mapsto 1. \end{aligned}$$

Find the boolean expression.

Solution 4.

$$=$$

Problem 5. Let $x_0, y_0 \in \{0,1\}$. Solve the system of boolean equations

$$x_{t+1} = x_t \oplus y_t, \qquad y_{t+1} = x_t \cdot y_t$$

where $x_0 = 0$, $y_0 = 1$ and $t = 0, 1, \ldots$. Here $\oplus$ denotes the XOR operation and $\cdot$ denotes the AND operation. First find the fixed points, i.e. solve $x \oplus y = x$, $x \cdot y = y$. Does the sequence x_t, y_t tend to a fixed point?

Solution 5. Testing the combinations $(0,0)$, $(0,1)$, $1,0)$, $(1,1)$ we obtain the fixed points $(0,0)$ and $(1,0)$. We obtain

$$x_1 = x_0 \oplus x_0 = 0 \oplus 1 = 1, \quad y_1 = x_0 \cdot y_0 = 0.$$

Thus we reach the fixed point $(1,0)$.

Problem 6. (i) Let $s_1(0), s_2(0), s_3(0) \in \{+1, -1\}$. Study the time-evolution $(t = 0, 1, 2, \ldots)$ of the coupled system of equations

$$\begin{aligned} s_1(t+1) &= s_2(t)s_3(t) \\ s_2(t+1) &= s_1(t)s_3(t) \\ s_3(t+1) &= s_1(t)s_2(t) \end{aligned}$$

for the eight possible initial conditions, i.e. (i) $s_1(0) = s_2(0) = s_3(0) = 1$, (ii) $s_1(0) = 1$, $s_2(0) = 1$, $s_3(0) = -1$, (iii) $s_1(0) = 1$, $s_2(0) = -1$, $s_3(0) = 1$, (iv) $s_1(0) = -1$, $s_2(0) = 1$, $s_3(0) = 1$, (v) $s_1(0) = 1$, $s_2(0) = -1$, $s_3(0) = -1$, (vi) $s_1(0) = -1$, $s_2(0) = 1$, $s_3(0) = -1$, (vii) $s_1(0) = -1$, $s_2(0) = -1$, $s_3(0) = 1$, (viii) $s_1(0) = -1$, $s_2(0) = -1$, $s_3(0) = -1$. Which of these initial conditions are fixed points?
(ii) Let $s_1(0), s_2(0), s_3(0) \in \{+1, -1\}$. Study the time-evolution ($t = 0, 1, 2, \ldots$) of the coupled system of equations

$$\begin{aligned} s_1(t+1) &= s_2(t)s_3(t) \\ s_2(t+1) &= s_1(t)s_2(t)s_3(t) \\ s_3(t+1) &= s_1(t)s_2(t) \end{aligned}$$

for the eight possible initial conditions, i.e. (i) $s_1(0) = s_2(0) = s_3(0) = 1$, (ii) $s_1(0) = 1$, $s_2(0) = 1$, $s_3(0) = -1$, (iii) $s_1(0) = 1$, $s_2(0) = -1$, $s_3(0) = 1$, (iv) $s_1(0) = -1$, $s_2(0) = 1$, $s_3(0) = 1$, (v) $s_1(0) = 1$, $s_2(0) = -1$, $s_3(0) = -1$, (vi) $s_1(0) = -1$, $s_2(0) = 1$, $s_3(0) = -1$, (vii) $s_1(0) = -1$, $s_2(0) = -1$, $s_3(0) = 1$, (viii) $s_1(0) = -1$, $s_2(0) = -1$, $s_3(0) = -1$. Which of these initial conditions are fixed points?

Solution 6.

=

Problem 7. Let $x_1(0), x_2(0), x_3(0) \in \{0, 1\}$ and let $\oplus$ be the XOR-operation. Study the time-evolution ($t = 01, 2, \ldots$) of the coupled system of equations

$$\begin{aligned} x_1(t+1) &= x_2(t) \oplus x_3(t) \\ x_2(t+1) &= x_1(t) \oplus x_3(t) \\ x_3(t+1) &= x_1(t) \oplus x_2(t) \end{aligned}$$

for the eight possible initial conditions, i.e. (i) $x_1(0) = x_2(0) = x_3(0) = 0$, (ii) $x_1(0) = 0$, $x_2(0) = 0$, $x_3(0) = 1$, (iii) $x_1(0) = 0$, $x_2(0) = 1$, $x_3(0) = 0$, (iv) $x_1(0) = 1$, $x_2(0) = 0$, $x_3(0) = 0$, (v) $x_1(0) = 0$, $x_2(0) = 1$, $x_3(0) = 1$, (vi) $x_1(0) = 1$, $x_2(0) = 0$, $x_3(0) = 1$, (vii) $x_1(0) = 1$, $x_2(0) = 1$, $x_3(0) = 0$, (viii) $x_1(0) = 1$, $x_2(0) = 1$, $x_3(0) = 1$. Which of these initial conditions are fixed points?

Solution 7.

=

Problem 8. Show that one Fredkin gate is sufficient to implement the XOR gate.

Solution 8. (i) Choosing $b = NOTc$ (equivalently $c = NOTb$) we find that the Fredkin gate yields

$$(a, b, c) \rightarrow (a,\, ORANDNOTabANDaNOTb,\, ORANDNOTacANDaNOTc) \equiv (a, a \oplus b, a \oplus c).$$

Thus we can apply the Fredkin gate to $(a, b, NOTb)$ and use the second bit to obtain $a \oplus b$ or equivalently apply the Fredkin gate to $(a, NOTc, c)$ and use the third bit to obtain $a \oplus c$. The question did not state that we can assume that $NOTb$ or $NOTc$ are available, which is required to obtain this result.

Problem 9. Show that $\overline{a \cdot b} = \overline{a} + \overline{b}$ using (i) truth tables and (ii) properties of boolean algebra (with $a + 1 = 1$).

Solution 9. The complement $NOTa \cdot b$ is defined by

$$(NOTa \cdot b) \cdot (a \cdot b) = 0$$

and

$$(NOTa \cdot b) + (a \cdot b) = 1.$$

We prove each of these properties for $NOTa + NOTb$.

$$(NOTa + NOTb) \cdot (a \cdot b) = (NOTa \cdot a) \cdot b + a \cdot (NOTb \cdot b) = 0 \cdot b + a \cdot 0 = 0 + 0 = 0.$$

Thus the first property holds. For the second property we have

$$\begin{aligned}
(NOTa + NOTb) + (a \cdot b) &= NOTa \cdot (b + NOTb) + NOTb + a \cdot b \\
&= NOTa \cdot b + a \cdot b + NOTb + NOTa \cdot NOTb \\
&= (NOTa + a) \cdot b + NOTb + NOTa \cdot NOTb \\
&= (b + NOTb) + (NOTa \cdot NOTb) \\
&= 1 + (NOTa \cdot NOTb).
\end{aligned}$$

We know that $a + 1 = 1$ (given, exercises) so that

$$(NOTa + NOTb) + (a \cdot b) = 1 + (NOTa \cdot NOTb) = 1$$

as required. Consequently

$$NOTa \cdot b = NOTa + NOTb.$$

Problem 10. Let $t = 0, 1, 2, \ldots$ and $x_t, y_t, z_t \in \{0, 1\}$. Solve

$$\begin{aligned}
x_{t+1} &= y_t \cdot z_t \\
y_{t+1} &= z_t + x_t \\
z_{t+1} &= x_t \oplus y_t
\end{aligned}$$

with the initial condition $x_0 = y_0 = z_0 = 1$. Discuss. Here $\cdot$ denotes the AND-operation, $+$ the OR-operation and $\oplus$ the XOR-operation. First find the fixed points of the map.

Solution 10.

Problem 11. A boolean function $f : \{\,0,1\,\}^n \to \{\,0,1\,\}$ can be written as

$$f(\mathbf{x}) = x_j \cdot f_{x_j}(\mathbf{x}) + \overline{x}_j \cdot f_{\overline{x}_j}(\mathbf{x})$$

where $j = 1, \ldots, n$, f_{x_j} denotes setting the variable x_j to 1 in f and $f_{\overline{x}_j}$ denotes setting the variable x_j to 0 in f. Apply this decomposition to $f(x_1, x_2) = x_1 \oplus x_2$.

Problem 12. Consider a bitstring of length 4 with two 0's and two 1's, for example `0110`. How many distinct permutations can be formed for such bitstrings? Order these bitstrings from smallest to largest. Recall that the least significant bit is on the right-hand side and counting starts from 0.

Solution 12.

Problem 13. Let $x_{1,t}, x_{2,t} \in \{0,1\}$ and $t = 0, 1, 2, \ldots$. Consider the map

$$x_{1,t+1} = x_{1,t} + x_{2,t}, \quad x_{2,t+1} = x_{1,t} \oplus x_{2,t}$$

where $+$ is the OR-operation and $\oplus$ is the XOR-operation. Find the fixed points of the map. Let $x_{1,0} = x_{2,0} = 1$. Does $(x_{1,t}, x_{2,t})$ tend to a fixed point?

Solution 13.

Problem 14. Consider a bitstring of length 4 with two 0's and two 1's, for example `0110`. How many distinct permutations can be formed for such bitstrings? Order these bitstrings from smallest to largest. Recall that the least significant bit is on the right-hand side and counting starts from 0.

Solution 14.

Problem 15. The boolean function $f : \{0,1\}^3 \to \{0,1\}$

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3 + \overline{x}_1 \cdot x_2 \cdot x_3 + \overline{x}_1 \cdot x_2 \cdot \overline{x}_3$$

is in sum-of-product form. Can the boolean function be reduced to

$$f(x_1, x_2, x_3) = x_2 \cdot (x_3 + \overline{x}_1)\,?$$

Prove or disprove.

Solution 15.

Chapter 3

Maps and Functions

Problem 1. The straight line *Hough transform* maps a line in $\mathbb{R}^2$ into a point in the Hough transform space. The polar definition of the Hough transform is based on the representation of the lines by the parameters (ρ, θ) via the equation

$$\rho = x_j \cos(\theta) + y_j \sin(\theta).$$

with $\rho \geq 0$ and $\theta \in [0, 2\pi)$. All points (x_j, y_j) of a given line correspond to a point (ρ, θ) in the Hough transform space. Any point (x_j, y_j) is mapped to a sinusoidal curve in the Hough transform space. Consider the two points $(x_0, y_0) = (1, 0)$ and $(x_1, y_1) = (0, 1)$ on a line. Find ρ, θ.

Solution 1. For the two points (x_0, y_0) and (x_1, y_1) we find the two equations

$$\rho = \cos(\theta), \qquad \rho = \sin(\theta).$$

Since $\rho = 1$ we find $1 = \tan\theta$ and therefore $\theta = \pi/4$. This means $\theta = 45°$. It follows that $\rho = \cos(\pi/4) = \sin(\pi/4) = 1/\sqrt{2}$. Consequently

$$(\rho, \theta) = (1/\sqrt{2}, \pi/4).$$

Thus ρ is the shortest distance between the straight line and the origin $(0, 0)$. The angle θ is the angle between the distance vector and the positive x-direction.

Problem 2. Let $f, g : \mathbb{R} \to \mathbb{R}$ be analytic functions and $n \geq 1$. Then

$$\int_a^b f^{(n)} g dx = f^{(n-1)} g \Big|_a^b - f^{(n-2)} g' \Big|_a^b + f^{(n-3)} g'' \Big|_a^b - \cdots (-1)^n \int_a^b f g^{(n)} dx.$$

Here $f^{(n)}$ denotes the n-th derivative. This identity is called *generalized integration by parts*. Let $\epsilon > 0$. Find

$$\int_0^1 e^{\epsilon x} x^n dx$$

using generalized integration by parts.

Solution 2.

Problem 3. Find a polynomial

$$p(x) = ax^4 + bx^3 + cx^2 + dx + e$$

which satisfies the conditions

$$p(0) = 0, \quad p(1) = 0, \quad p(1/2) = 0$$

$$p(1/4) = 1, \qquad p(3/4) = 1/2.$$

Solution 3. From $p(0) = 0$ we obtain $e = 0$. From the other four conditions we find the system of four linear equations with four unkowns

$$\begin{aligned} 0 &= a + b + c + d \\ 0 &= \frac{a}{16} + \frac{b}{8} + \frac{c}{4} + \frac{d}{2} \\ 1 &= \frac{a}{256} + \frac{b}{64} + \frac{c}{16} + \frac{d}{4} \\ \frac{1}{2} &= \frac{81a}{256} + \frac{27b}{64} + \frac{9c}{16} + \frac{3d}{4}. \end{aligned}$$

The solution is

$$a =, \quad b =, \quad c =, \quad d = .$$

Problem 4. Let A, B and C be arbitrary non-empty sets and let $f : A \to B$ and $g : B \to C$. The composite function of f and g is the function

$$g \circ f \, : \, A \to C, \quad (g \circ f)(x) = g(f(x)).$$

Notice that $g \circ f$ reads from right to left; it means first apply f, then apply g to the result. Note that function composition is associative.
(i) Let $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$, and $g : \mathbb{R} \to \mathbb{R}$, $g(x) = 3x - 1$. Find $g \circ f$ and $f \circ g$.
(ii) Write a C++ program which implements these compositions with x of data type `double`.

Solution 4. (i) The function $g \circ f$ is found as follows $g \circ f : \mathbb{R} \to \mathbb{R}$,

$$(g \circ f)(x) = g(f(x)) = g(x^2) = 3x^2 - 1.$$

The function $f \circ g$ is obtained in a similar manner $f \circ g : \mathbb{R} \to \mathbb{R}$,

$$(f \circ g)(x) = f(g(x)) = f(3x - 1) = (3x - 1)^2.$$

(ii) The C++ program is as follows.

```
// composition.cpp

#include <iostream>
using namespace std;

double f1(double x) { return x*x; }

double g1(double (*f)(double),double x) { return 3.0*f(x) - 1.0; }

double f2(double x) { return 3.0*x - 1.0; }

double g2(double (*f)(double),double x) { return f(x)*f(x); }

int main(void)
{
 double x;
 for(x=0.0;x<=2.0;x+=0.1)
 { cout << "g1(" << x << ") = " << g1(f1,x) << endl; }
 double y;
 for(y=0.0;y<=2.0;y+=0.1)
 { cout << "g2(" << y << ") = " << g2(f2,y) << endl; }
 return 0;
}
```

Problem 5. Let f_1, f_2 be continuous functions over an interval $[a, b]$. Then we have the identities

$$\min(f_1, f_2) \equiv \frac{1}{2}(f_1 + f_2 - |f_1 - f_2|)$$
$$\max(f_1, f_2) \equiv \frac{1}{2}(f_1 + f_2 + |f_1 - f_2|).$$

Write a C++ program that finds the min and max for two given continuous functions f_1 and f_2 using the function

```
void minmax(double (*f1)(double),double (*f2)(double),
            double x,double& min,double& max)
```

where x is the function parameter. Apply it to the sine function and cosine funtion in the interval $[0, 2]$.

Solution 5.

```
// minmax.cpp

#include <iostream>
#include <cmath>
using namespace std;

void minmax(double (*f1)(double),double (*f2)(double),
                    double x,double& min,double& max)
{
 double t1 = f1(x) + f2(x);
 double t2 = fabs(f1(x)-f2(x));
 min = 0.5*(t1-t2); max = 0.5*(t1+t2);
}

int main(void)
{
 double min, max;
 for(double x=0.0;x<=2.0;x=x+0.1)
 {
 minmax(sin,cos,x,min,max);
 cout << "x = " << x << " " << "min = " << min << endl;
 cout << "x = " << x << " " << "max = " << max << endl;
 }
 return 0;
}
```

Problem 6. Consider the two membership functions $f : \mathbb{R} \to [0, 1]$, $g : \mathbb{R} \to [0, 1]$ in fuzzy logic

$$f(x) = e^{-x^2/2}, \qquad g(x) = 1/(1 + e^{-x}).$$

Write a C++ program that finds the algebraic sum (page 524, Nonlinear Workbook 5th edition).

Solution 6.

Problem 7. What is the output of the following C++ program

```
// fcomposition.cpp

#include <iostream>
#include <cmath>
```

```
using namespace std;

double f(double x) { return x*x; }
double g(double x) { return 3.0*x-1.0; }

double comp(double (*f)(double),double (*g)(double),double x)
{ f(g(x)); }

int main(void)
{
  double x = 2.5;
  cout << "f(" << x << ") = " << f(x) << endl;
  cout << "g(" << x << ") = " << g(x) << endl;
  cout << comp(f,g,x) << endl;
  return 0;
}
```

Solution 7.

Problem 8. Let $\mathbb{N}_0$ be the set of natural numbers including 0. The *Cantor pairing function* $f : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0$ is defined by

$$f(x, y) = y + \frac{1}{2}(x + y)(x + y + 1).$$

(i) Find the inverse function, i.e. given $s = f(x, y)$ find x and y. Set

$$a := x + y, \quad b := \frac{1}{2}(a^2 + a).$$

(ii) Give a C++ implementation utilizing `Verylong` of SymbolicC++.

Solution 8. (i) We have $s = b + y$. Solving the quadratic equation $a^2 + a = b$ yields

$$a = \frac{1}{2}(\sqrt{8b + 1} - 1).$$

Since

$$b \leq s = b + y < b + (a + 1) = \frac{1}{2}((a + 1)^2 + (a + 1))$$

we obtain

$$a = \lfloor \frac{\sqrt{8s + 1} - 1}{2} \rfloor.$$

This means we find x, y via

$$y = s - \frac{1}{2}(a^2 + a), \qquad x = a - y.$$

(ii) The C++ implementation is

```
// CantorPairing.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
  int s = 8;
  int h1 = 8*s + 1;
  double h2 = (double) h1;
  double h3 = sqrt(h2);
  double h4 = (h3-1.0)/2.0;
  double h5 = floor(h4);
  int a = (int) h5;
  int y = s - (a*a+a)/2;
  int x = a - y;
  cout << "x = " << x << endl;
  cout << "y = " << y << endl;
  return 0;
}
```

```
// CantorPairing.cpp

#include <iostream>
#include <cmath>
using namespace std;

int f(int x,int y)
{
 return ((x+y)*(x+y+1))/2 + y;
}

void fI(int z,int* p)
{
  int w = floor(((sqrt(8.0*z+1.0)-1.0))/2.0);
  int t = (w*w+w)/2;
  p[1] = z-t;
  p[0] = w-p[1];
}

int main(void)
{
  int x = 3; int y = 5;
  int r = f(x,y);
  cout << "r = " << r << endl;
  int* p = new int[2];
  int z = 23;
```

```
  fI(z,p);
  cout << "p[0] = " << "x = " << p[0] << endl;
  cout << "p[1] = " << "y = " << p[1] << endl;
  delete[] p;
  return 0;
}
```

Problem 9. Consider the mathematical expression

$$\sin(b) + a * b \underbrace{+}_{} c * d + (a - b).$$

Write this mathematical expression as a binary tree with the root indicated by the brace. Then evaluate this binary tree from bottom to top with the values $a = 2$, $b = \pi/2$, $c = 4$, $d = 1$.

voluntary. An alternative to represent a mathematical expression as tree is multiexpression programming (see next page). Use multiexpression programming to evaluate the expression.

Solution 9.

Problem 10. (i) Let $r > 0$ (fixed) and $x > 0$. Consider the map

$$f_r(x) = \frac{1}{2}\left(x + \frac{r}{x}\right)$$

or written as difference equation

$$x_{t+1} = \frac{1}{2}\left(x_t + \frac{r}{x_t}\right), \qquad t = 0, 1, 2, \ldots \quad x_0 > 0.$$

Find the fixed points of f_r. Are the fixed points stable?
(ii) Let $r = 3$ and $x_0 = 1$. Find $\lim_{t\to\infty} x_t$. Discuss.

Solution 10. (i) From $f_r(x^*) = x^*$ we find the only fixed point $x^* = \sqrt{r}$. Now

$$\frac{df_r(x)}{dx} = \frac{1}{2} - \frac{1}{2}\frac{r}{x^2}.$$

Thus $df_r(x = x^*) = 0$. Thus the fixed point x^* is stable.
(ii) Obviously x_t tends to $\sqrt{3}$.

Problem 11. Let A be a given 3×3 matrix over $\mathbb{R}$ with $\det(A) \neq 0$. Is the transformation

$$x'(x, y) = \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + a_{33}}$$
$$y'(x, y) = \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + a_{33}}$$

invertible? If so find the inverse.

Solution 11.

Problem 12. Let N_1, N_2, N_3 be positive integers. Let $n_1 = 0, 1, \ldots, N_1 - 1$, $n_2 = 0, 1, \ldots, N_2 - 1$, $n_3 = 0, 1, \ldots, N_3 - 1$. There are $N_1 \cdot N_2 \cdot N_3$ points and the points (n_1, n_2, n_3) are a subset of $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$ and can be mapped one-to-one onto a subset of $\mathbb{N}_0$

$$j(n_1, n_2, n_3) = (n_1 N_2 + n_2) N_3 + n_3$$

where $j = 0, 1, \ldots, N_1 \cdot N_2 \cdot N_3 - 1$. Consider first the case $N_1 = N_2 = N_3 = 2$.

Solution 12. For $N_1 = N_2 = N_3 = 2$ we have

$$(0,0,0) \leftrightarrow 0, \quad (0,0,1) \leftrightarrow 1, \quad (0,1,0) \leftrightarrow 2, \quad (0,1,1) \leftrightarrow 3$$

$$(1,0,0) \leftrightarrow 4, \quad (1,0,1) \leftrightarrow 5, \quad (1,1,0) \leftrightarrow 6, \quad (1,1,1) \leftrightarrow 7.$$

For general N_1, N_2, N_3 the inverse map is given by

$$n_1 = \left\lfloor \frac{t}{N_2} \right\rfloor, \quad n_2 = t - N_2 \left\lfloor \frac{t}{N_2} \right\rfloor, \quad n_3 = j - N_3 \left\lfloor \frac{j}{N_3} \right\rfloor$$

where $t = \lfloor j/N_3 \rfloor$.

Problem 13. Give an implementation of the function

$$\delta^{i_1 i_2 \ldots i_k}_{j_1 j_2 \ldots j_k} := \frac{1}{k!} \sum_{\pi \in S_k} \operatorname{sgn}(\pi) \delta^{i_{\pi(1)}}_{j_1} \delta^{i_{\pi(2)}}_{j_2} \cdots \delta^{i_{\pi(k)}}_{j_k}.$$

For example

$$\delta^{ij}_{k\ell} = \frac{1}{2}(\delta^i_k \delta^j_\ell - \delta^i_\ell \delta^j_k).$$

Solution 13.

Problem 14. Let $i_0, i_1, \ldots, i_{k-1} \in \mathbb{N}_0$. Given a vector $(i_0, i_1, \ldots, i_{k-1})$. Give an implementation of the function

$$\zeta_{i_0, i_1, \ldots, i_{k-1}} = \begin{cases} -1 & \text{for} \quad i_0 > i_1 > \cdots > i_{k-1} \\ 1 & \text{for} \quad i_0 < i_1 < \cdots < i_{k-1} \\ 0 & \text{otherwise} \end{cases}$$

Solution 14.

=

Problem 15. Let $n_1, n_2, n_3 \in \mathbb{N}$. Consider the equation

$$\frac{1}{n_1} + \frac{1}{n_2} + \frac{1}{n_3} = \frac{1}{2}.$$

Write a SymbolicC++ program utilizing the class Rational and Verylong to test whether there are solutions for $n_1, n_2, n_3 \in \{1, 2, \ldots, 50\}$.

Solution 15. We find the six solutions

$$(3, 7, 42),\ \ (3, 8, 24),\ \ (3, 9, 18),\ \ (3, 10, 15),\ \ (4, 5, 20),\ \ (4, 6, 12).$$

Problem 16. Let N_1, N_2 be nonnegative integers and $N = N_1 + N_2$. Consider the function

$$f(N_1, N) = \frac{N!}{N_1! N_2!} \equiv \frac{N!}{N_1!(N - N_1)!}$$

Let $N = 10$. Find the minima and maxima of the function $f(N_1, 10)$.

Solution 16.

Problem 17. Consider the alphabet

$$\Sigma :=< x_1, x_2 >$$

and the morphism (map)

$$\sigma(x_1) = x_2, \qquad \sigma(x_2) = x_1 x_2.$$

For example starting with $x_2 x_1 x_1 x_2 x_2 x_1 x_1 x_1$ we obtain

$$x_1 x_2 x_2 x_2 x_1 x_2 x_1 x_2 x_2 x_2 x_2.$$

Write a C++ program that implements this map. Describe the connection with the Fibonacci numbers given by the recursion relation

$$y_{n+2} = y_{n+1} + y_n, \quad n = 0, 1, 2, \ldots$$

and $y_0 = y_1 = 1$,

Solution 17.

Problem 18. Show that the $(2, 2)$ *Padé approximant* of the cosine function is given by

$$\cos(x) \approx \frac{12 - 5x^2}{12 + x^2}.$$

Solution 18. SCandPP analysis

Chapter 4

Number Manipulations

Problem 1. Apply the *Chinese remainder theorem* to solve the set of equations

$$\begin{aligned} x &\equiv 7 \pmod 8 \\ x &\equiv 2 \pmod 9 \\ x &\equiv -1 \pmod 5. \end{aligned}$$

Chinese remainder theorem. Suppose that the positive integers m_1, m_2, ..., m_t are relatively prime in pairs; that is, $\gcd(m_i, m_j) = 1$ if $i \neq j$, $1 \leq i, j \leq t$. Let b_1, b_2, ..., b_t be arbitrary integers. Then the congruences

$$\begin{aligned} x &\equiv b_1 \pmod{m_1} \\ x &\equiv b_2 \pmod{m_2} \\ &\vdots \\ x &\equiv b_t \pmod{m_t} \end{aligned}$$

have a simultaneous solution. Moreover, the simultaneous solution is unique modulo $m_1 m_2 \cdots m_t$. That is, if y is another solution, then $x \equiv y \pmod{m_1 m_2 \cdots m_t}$.

To find x we write it in the form

$$x = y_1 b_1 + \cdots + y_t b_t$$

where $y_1 \equiv 1 \pmod{m_1}$ and $y_1 \equiv 0 \pmod{m_i}$ $(2 \leq i \leq t)$, $y_2 \equiv 1 \pmod{m_2}$ and $y_2 \equiv 0 \pmod{m_i}$ $(i = 1, 3, 4, \ldots, t)$ and similarly for $y_3, \ldots, y_t$. To have $y_1 \equiv 0 \pmod{m_i}$ $(2 \leq i \leq t)$ we must have $m_2 m_3 \cdots m_t | y_1$, since the m_i are

relative prime in pairs. Thus, in general, set

$$m_i' = \frac{m_1 m_2 \cdots m_t}{m_i}.$$

Then $\gcd(m_i, m_i') = 1$ since $m_1, m_2, \ldots, m_t$ are relatively prime in pairs. Thus m_i' has an arithmetic inverse $m_i'^* \bmod m_i$, i.e.

$$m_i'^* m_i' = 1 \,(\mathrm{mod}\, m_i).$$

We set $y_i = m_i'^* m_i'$ and correspondingly set

$$x = m_1'^* m_1' b_1 + \cdots + m_t'^* m_t' b_t.$$

We have $x \equiv b_1 \,(\mathrm{mod}\, m_1)$ since for $2 \le i \le t$ we have $m_1 | m_i'$ so that $m_i'^* m_i' b_i \equiv 0 \,(\mathrm{mod}\, m_1)$ for $2 \le i \le t$. We also have $m_1'^* m_1' \equiv 1 \,(\mathrm{mod}\, m_1)$ so that $m_1'^* m_1' b_1 \equiv b_1 \,(\mathrm{mod}\, m_1)$. Thus

$$x \equiv b_1 + 0 + \cdots + 0 \equiv b_1 \,(\mathrm{mod}\, m_1).$$

It follows similarly that $x = b_i \,(\mathrm{mod}\, m_i)$ for all i, $1 \le i \le t$.

Solution 1. We have $b_1 = 7$, $b_2 = 2$, $b_3 = -1$ and $m_1 = 8$, $m_2 = 9$, $m_3 = 5$. Thus

$$m_1' = 9 \cdot 5 = 45, \quad m_2' = 8 \cdot 5 = 40, \quad m_3' = 8 \cdot 9 = 72.$$

Therefore

$$45 m_1'^* \equiv 1 \,(\mathrm{mod}\, 8), \quad 40 m_2'^* \equiv 1 \,(\mathrm{mod}\, 9), \quad 72 m_3'^* \equiv 1 \,(\mathrm{mod}\, 5)$$

so that

$$m_1'^* \equiv 5 \,(\mathrm{mod}\, 8), \quad m_2'^* \equiv -2 \,(\mathrm{mod}\, 9), \quad m_3'^* \equiv 3 \,(\mathrm{mod}\, 5).$$

We therefore find

$$x = 5 \cdot 45 \cdot 7 + (-2) \cdot 40 \cdot 2 + 3 \cdot 72 \cdot (-1) = 1199 \equiv 119 \,(\mathrm{mod}\, 360)$$

where $360 = 8 \cdot 9 \cdot 5$. Thus $x = 119$ satisfies the congruences.

Problem 2. Let p and q be prime numbers with $p, q \ge 3$ and $p \ne q$. Let $n = pq$. Assume that $d, e \in \mathbb{N}$ be two integer numbers with the properties

$$\begin{gathered} 3 < e < (p-1)(q-1) \\ de = 1 \bmod (p-1)(q-1) \\ \gcd(e, n) = \gcd(e, (p-1)(q-1)) = 1. \end{gathered}$$

With these properties we can prove that for $M \in \{\, 0, 1, 2, \ldots, n-1 \,\}$ the definition

$$C := M^e \bmod n$$

yields $M = C^d \bmod n$. Consequently

$$C = C^{ed} \bmod n$$

(i) Let $s \in \mathbb{N}$ such that $C = C^{ed} \bmod n$. Show that

$$M = C^s \bmod n.$$

(ii) Show how the order r of C under modulo n arithmetic can be used to obtain a linear diophantine equation for s.
(iii) Let $p = 3$, $q = 17$, $e = 5$ and $M = 10$. Find s and d.

Solution 2. (i) We have

$$C^s \equiv (M^e)^s \equiv ((C^d)^e)^s \equiv (C^{es})^d \equiv C^d \equiv M \bmod n.$$

(ii) We note that

$$C^{es-1} \equiv 1 \bmod n.$$

If $\gcd(C, n) \neq 1$, then $s = p$ or $s = q$. If $\gcd(C, n) = 1$ we find that $es - 1$ is a multiple of the order r of C under modulo n arithmetic. Then we obtain r from the periodicity of the sequence

$$C^0 = 1,\, C^1,\, C^2,\, \ldots\, C^r \equiv 1,\; C^{r+1} \equiv C,\, C^2,\, \ldots \bmod n.$$

Since $es - 1$ is a multiple of r we have $es - 1 = kr$ for some $k \in \mathbb{Z}$, or $es - kr = 1$ which is a linear diophantine equation. This equation can be solved in polynomial time for non-negative s. If the algorithm yields negative s we simply use the relation

$$e(s + r) - (k + e)r = 1$$

i.e., we add r to s until we find a positive result.
(iii) We have $n = pq = 51$. Thus we find

$$\begin{aligned} C &:= 10^5 \bmod 51 = 10\,(100 \bmod 51)^2 \bmod 51 \\ &= 10\,(49)^2 \bmod 51 = 40 \end{aligned}$$

and

$$\begin{aligned} C^2 \bmod 51 &= 19 & C^3 \bmod 51 &= 46 \\ C^4 \bmod 51 &= 4 & C^5 \bmod 51 &= 7 \\ C^6 \bmod 51 &= 25 & C^7 \bmod 51 &= 31 \\ C^8 \bmod 51 &= 16 & C^9 \bmod 51 &= 28 \\ C^{10} \bmod 51 &= 49 & C^{11} \bmod 51 &= 22 \\ C^{12} \bmod 51 &= 13 & C^{13} \bmod 51 &= 10 \\ C^{14} \bmod 51 &= 43 & C^{15} \bmod 51 &= 37 \\ C^{16} \bmod 51 &= 1. \end{aligned}$$

Thus $r = 16$. Consequently we need to solve the linear diophantine equation

$$5s - 16k = 1.$$

Obviously a solution is given by $s = -3$ and $k = -1$. By translation we find a second solution $s = 13$ and $k = 4$. It is simple to verify that $4^{13} \bmod 51 = 10$. To find d we solve the linear diophantine equation

$$de - k_2(p-1)(q-1) = 1$$

i.e. $5d - 32k_2 = 1$, to obtain $d = 13$ and $k_2 = 2$. Thus we find in this case $d = s$.

Problem 3. Let $\mathbb{F}$ be a field ($\mathbb{F} = \mathbb{R}, \mathbb{C}$). Consider polynomials. The *division algorithm* is as follows. Let g be a nonzero polynomial in $\mathbb{F}[x]$. Then every p in $\mathbb{F}[x]$ can be written as

$$p = qg + r$$

where q anbd r are in $\mathbb{F}[x]$, and either $r = 0$ or $\deg(r) < \deg(g)$. Furthermore q and r are unique and can be found by the following algorithm

```
input:   p, g
output: q, r
q := 0; r := p
while(r <> 0) and LT(g) divides LT(r)) do
 q := q + LT(r)/LT(g)
 r := r - (LT(r)/LT(g))
```

where `LT` is the leading term, i.e. the term with the highest degree. Apply the division algorithm to

$$p(x) = x^4 - 1, \qquad g(x) = x^3 - x^2 + x - 1.$$

Write a C++ program using SymbolicC++ that implements the algorithm.

Solution 3. We have

$$=$$

Problem 4. Consider the bijective spiral map on page 79 (problem 19). Can we find an explicit expression for f? Could a polynomial ansatz work

$$f(x,y) = \sum_{i,j=0}^{N} c_{ij} x^i y^j, \qquad (x,y) \in \mathbb{Z} \times \mathbb{Z}.$$

Solution 4.

Problem 5. What is the output of the following C++ program

```
// successor.cpp

#include <iostream>
using namespace std;

template<int n> class number
{
  private:
    number<n-1> predecessor;
  public:
    number<n+1> successor(void)
    { return number<n+1>(); }

  ostream& output(ostream& o)
  { o << "{" << predecessor << "}"; return o; }
}; // end class number

class number<0>
{
  public:
    number<1> successor(void)
    { return number<1>(); }
}; // end class number<0>

template <int n> ostream& operator << (ostream& o,number<n> n)
{  return n.output(o); }

ostream& operator << (ostream& o,number<0> n)
{ o << "{ }"; return o; }

int main(void)
{
  number<0> zero;
  cout << zero  << endl;
  cout << zero.successor() << endl;
  cout << zero.successor().successor() << endl;
  return 0;
}
```

Solution 5. The output is

```
{  }
{{  }}
{{{  }}}
```

Problem 6. The following program uses the `Verylong` class of SymbolicC++. What is the program doing?

```
// wormell.cpp

#include <iostream>
#include "verylong.h"
using namespace std;

int main(void)
{
 Verylong two("2");
 for(Verylong x("3");x<=Verylong("100");x+=2)
 {
 Verylong p("1");
 Verylong t = x/two;
 for(Verylong a("2");a<=t;a++)
 { for(Verylong b("2");b<=t;b++) { p = p*(x-a*b); } }
 cout << "x = " << x << " " << "p = " << p << endl;
 } // end x for loop
 return 0;
}
```

Solution 6. The program implements Wormell's formula, i.e.

$$B(x) = \prod_{a=2}^{x} \prod_{b=2}^{x} (x - ab)^2 = \begin{cases} \text{a positive integer, if } x \text{ is prime} \\ 0 \text{ if x is composite} \end{cases}$$

Since the positive integer is very large the data type `int` would be to small.

Problem 7. What is the output of the following program?

```
// cypher.cpp

#include <iostream>
#include <string>
using namespace std;

int main(void)
{
  string input = "PLEASE CONFIRM RECEIPT 471";
  string output;
  char t = 3;

  for(int i=0;i<input.length();i++)
  {
  if(('a' <= input[i]) && (input[i] <= 'z'))
       output += (input[i] - 'a' + t)%26 + 'a';
  else if(('A' <= input[i]) && (input[i] <= 'Z'))
       output += (input[i] - 'A' + t)%26 + 'A';
```

```
  else if(('0' <= input[i]) && (input[i] <= '9'))
       output += (input[i] - '0' + t)%10 + '0';
  else output += input[i];
  }
  cout << "output = " << output << endl;
  return 0;
}
```

Solution 7. The output is

```
SOHDVH FRQILUP UHFHLSW 704
```

Problem 8. Let n be a positive integer. We define the set $\mathbb{Z}_n$ as the set of nonnegative integers less than n

$$\mathbb{Z}_n := \{\, 0, 1, 2, \ldots, (n-1) \,\}.$$

This is referred to as the set of residues modulo n. If we perform modular arithmetic within this set the following properties hold

$$\begin{aligned}
\text{Commutative laws} \quad & (w + x) \bmod n = (x + w) \bmod n \\
& (w * x) \bmod n = (x * w) \bmod n \\
\text{Associative laws} \quad & ((w + x) + y) \bmod n = (w + (x + y)) \bmod n \\
& ((w * x) * y) \bmod n = (x * (w * y)) \bmod n \\
\text{Distributive law} \quad & (w * (x + y)) \bmod n = ((w * x) + (w * y)) \bmod n \\
\text{Identities} \quad & (0 + w) \bmod n = w \bmod n \\
& (1 * w) \bmod n = w \bmod n
\end{aligned}$$

and we have an additive inverse $(-w)$, i.e. for each $w \in \mathbb{Z}_n$ there exists a z such that $w + z = 0 \bmod n$. Note that

```
if (a*b) = (a*c) mod n then b = c mod n if a is relatively
prime to n
```

Write a C++ program using templates that implements modular arithmetic.

Solution 8.

Problem 9. The change problem is as follows. Convert some amount of money M into given denominations, using the smallest possible number of coins. This means the input is an amount of money M and an array of d denominations $\mathbf{c} = (c_0, c_1, \ldots, c_{d-1})$ in decreasing order of value, i.e. $c_0 > c_1 > \cdots > c_{d-1}$, where $c_{d-1} = 1$. The output is an array of d integers $i_0, i_1, \ldots, i_{d-1}$ such that

$$c_0 i_0 + c_1 i_1 + \cdots + c_{d-1} i_{d-1} = M$$

and $i_0 + i_1 + \cdots + i_{d-1}$ is a small as possible. The pseudocode is as follows

smallestNumberOfCoins = M

for each $(i_0, \ldots, i_{d-1})$ from $(0, \ldots, 0)$ to $(M/c_0, \ldots, M/c_{d-1})$

  valueOfCoins = $\sum_{k=0}^{d-1} i_k c_k$

    if valueOfCoins = M

      numberOfcoins = $\sum_{k=0}^{d-1} i_k$

      if numberOfCoins < smallestNumberOfCoins

        smallestNumberOfCoins = numberOfCoins

        bestChange = $(i_0, i_1, \ldots, i_{d-1})$

return array bestChange

Write the pseudocode as a C++ program.

Solution 9.

Problem 10. Write a C++ program using `Verylong` of SymbolicC++ that finds all integer solutions of

$$2437x + 51329y = 1 \qquad x, y \in \mathbb{Z}$$

in the range $x, y \in [-100000, 100000]$.

Solution 10. We find $x = 20599$ and $y = -978$. Note the the greatest common divisor of 2437 and 51329 is $\gcd(2437, 51329) = 1$.

Problem 11. We define a mapping from the natural numbers $\mathbb{N}_0$ to sets as

$$0 \to \{\ \}, \qquad n + 1 \to \{\, n \,\}.$$

Give a C++ implementation of this representation of natural numbers.

Solution 11.

```
// mapnatural.cpp

#include <iostream>
```

```
using namespace std;

class Zero
{
 public:
  ostream &output(ostream &o) const { return o << "{ }"; }
};

template <class T> class Successor
{
 private:
  T predecessor;
 public:
  Successor(const T &t) : predecessor(t) {}
  ostream &output(ostream &o) const
   {
   o << "{"; predecessor.output(o); o << "}";
   return o;
   }
};

template <class T>
Successor<T> successor(const T &n) { return Successor<T>(n); }

ostream &operator<<(ostream &o,Zero num)
{ return num.output(o); }

template <class T>
ostream &operator<<(ostream &o,Successor<T> num)
{ return num.output(o); }

int main(void)
{
 Zero zero;
 cout << zero                                    << endl
      << successor(zero)                         << endl
      << successor(successor(zero))              << endl
      << successor(successor(successor(zero))) << endl;
 return 0;
}
```

Problem 12. Let m and n be positive integers. We define the map $\mathbb{N}\times\mathbb{N}\to\mathbb{N}$

$$m\blacklozenge n := \frac{\text{the lowest common multiple of } m \text{ and } n}{\text{the highest common factor of } m \text{ and } n}.$$

For example

$$12\blacklozenge 30 = \frac{60}{6} = 10.$$

Is the composition associative, i.e.

$$(m\blacklozenge n)\blacklozenge p = m\blacklozenge(n\blacklozenge p)\,?$$

Write a C++ program using templates (so that `Verylong` of SymbolicC++ can also be used) that implements this composition.

Solution 12. The composition is not associative. An example is

$$(12\blacklozenge 30)\blacklozenge 14 = 10\blacklozenge 14 = 35.$$

However

$$12\blacklozenge(30\blacklozenge 14) = 12\blacklozenge 105 = 140.$$

Problem 13. Consider the following arithmetic problem

```
  # # * #
 --------
  # #
+ # #
 ----
  # #
```

where $*$ denotes multiplication and $+$ denotes addition. Each `#` should be one of the digit 1,2,3,4,5,6,7,8,9, where the condition is that each digit occurs only once. Write a C++ program that finds all the solutions.

Solution 13.

Problem 14. Let n be a positive integer. There are exactly as many irreducible representations of the permutation group S_n (order of S_n is $n!$) as there are *partitions* $\{p_j\}$ of n

$$\sum_{j=1}^{n} p_j = n, \qquad p_1 \geq p_2 \geq \cdots \geq p_n \geq 0.$$

For example for $n = 4$ we have 5 partitions with

$$4000, \quad 3100, \quad 2200, \quad 2110, \quad 1111.$$

The number of partitions is given by $p(1, k)$, where $p(k, n)$ represents the number of partitions of n using only natural numbers at least as large as k. A recursion relation for $p(k, n)$ is given by

$$p(k,n) = \begin{cases} 0 & \text{if} \quad k > n \\ 1 & \text{if} \quad k = n \\ p(k+1,n) + p(k+1,n) + p(k,n-k) & \text{otherwise} \end{cases}$$

(i) Give a C++ implementation of this recursion.
(ii) Give a C++ implementation that finds the partitions for a given n without the trailing 0's.

Solution 14. (i)

```
// NoofPartitions.cpp

#include <iostream>
using namespace std;

int p(int k,int n)
{
 if(k > n) return 0;
 if(k==n) return 1;
 else return p(k+1,n) + p(k,n-k);
}

int main(void)
{
  int r1 = p(2,7);
  cout << "r1 = " << r1 << endl;
  int r2 = p(1,10);
  cout << "r2 = " << r2 << endl;
  return 0;
}
```

(ii) The implementation is

```
// partition.cpp

#include <iostream>
using namespace std;

int main(void)
{
 int n = 7;
 int* T = new int[n+1];
 int m, h;
 T[0] = -1;
 int j; int r;
 for(j=1;j<=n;j++) T[j] = 1;
 for(j=1;j<=n;j++) cout << T[j] << endl;
 cout << endl;
 T[1] = 2; h = 1; m = n-1;
 for(j=1;j<=m;j++) cout << T[j] << endl;
 cout << endl;

 while(T[1] != n)
```

```
{
if((m-h) > 1) { h = h + 1; T[h] = 2; m = m - 1; }
else { j = m - 2;
  while(T[j]==T[m-1]) { T[j] = 1; j = j-1; }
  h = j+1; T[h] = T[m-1] + 1;
  r = T[m] + T[m-1]*(m-h-1);
  T[m] = 1;
  if((m-h) > 1) T[m-1] = 1;
  m = h+r-1; }
   for(j=1;j<=m;j++) cout << T[j] << endl;
   cout << endl;
  }
return 0;
}
```

Problem 15. Consider the sequence (*Ballot numbers*)

$$B_0 = B_1 = 1,$$
$$B_L = B_{L-1} + \sum_{\ell=0}^{L-2} B_\ell B_{L-2-\ell}, \qquad L = 2, 3, \ldots.$$

(i) Give a C++ implementation of this recursion.
(ii) Show that the generating function

$$B(x) = \sum_{L=0}^{\infty} B_L x^L$$

is given by

$$x^2 (B(x))^2 - (1-x)B(x) + 1 = 0.$$

Solution 15.

Problem 16. A *perfect number* is a natural number with the properties that the sum of the factors gives twice the number, for example

$$2 \cdot 6 = 1 + 2 + 3 + 6$$

so that 6 is a perfect number. A *Mersenne prime* is a Mersenne number $2^n - 1$, where n is chosen such that $2^n - 1$ is prime (for example $n = 5$).
(i) Show that $2^{n-1}(2^n - 1)$ is perfect. For example for $n = 5$ we have 496 with

$$496 = 248 + 124 + 62 + 31 + 16 + 8 + 4 + 2 + 1.$$

(ii) Prove that if $2^{n-1}p$ is perfect for p prime then p is a Mersenne prime and $p = 2^n - 1$.

(iii) Prove that for any even perfect number q there exists $n \in \mathbb{N}$ such that $q = 2^{n-1}(2^n - 1)$.

Solution 16. (i) Since $2^n - 1$ is the prime, the factors of $2^{n-1}(2^n - 1)$ are

$$1,\, 2,\, 4,\, \ldots,\, 2^{n-1},\, 2^n - 1,\, 2(2^n - 1),\, 4(2^n - 1), \ldots,\, 2^{n-1}(2^n - 1).$$

Summing the factors we find

$$\sum_{j=0}^{n-1} 2^j + \sum_{j=0}^{n-1} 2^j(2^n - 1) = 2^n \sum_{j=0}^{n-1} 2^j = 2^n(2^n - 1) = 2 \cdot 2^{n-1}(2^n - 1).$$

Thus $2^{n-1}(2^n - 1)$ is perfect for $2^n - 1$ prime.
(ii) In the same way the factors of $2^{n-1}p$ are

$$1,\, 2,\, 4,\, \ldots,\, 2^{n-1},\, p,\, 2p,\, 4p, \ldots,\, 2^{n-1}p.$$

Summing the factors we find

$$\sum_{j=0}^{n-1} 2^j + \sum_{j=0}^{n-1} 2^j p = (p+1) \sum_{j=0}^{n-1} 2^j = (p+1)(2^n - 1) = 2 \cdot 2^{n-1}p.$$

Solving for p from the equation

$$(p+1)(2^n - 1) = 2^n p$$

yields $p = 2^n - 1$.

Problem 17. Consider the one-dimensional map (logistic map) $f : [0,1] \to [0,1]$

$$f(x) = 4x(1 - x).$$

A computational analysis using a finite state machine with base 2 arithmetic in fixed point operation provides one-dimensional maps with a lattice of 2^N sites labeled by numbers

$$x = \sum_{j=1}^{N} \frac{\epsilon_j}{2^j}, \qquad \epsilon_j \in \{\, 0, 1 \,\}$$

and N defines the machine's precision. Consider $N = 8$ bits and $x = 1/8$. Calculate the orbit $f(x)$, $f(f(x))$, $f(f(f(x)))$, ... with this precison. Discuss.

Solution 17. switch to 8 bits The bit representation of $1/8$ with $N = 6$ is `001000`. Now $f(1/8) = 7/16$ with the bit representation `011100` with no rounding errors. Next we have $f(f(1/8)) = 63/64$ with the bit representation `111111`. Again no rounding error occurs. Next we have

$$f(f(f(1/8))) = \frac{63}{1024} = \frac{2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0}{2^{10}}$$

with the bit representation `0000111111`. Since we only have 6 bits disposable we have to truncate the bitstring to `000011` which represents the number 3/64. Using 3/64 we find next in the sequence 183/1024 with the bit representation `0010110111`. We again have to truncate to `001011` which represents 11/64. Proceeding in this manner we find next 9/16 and then 63/64. Thus we obtain a 4-cycle after only 6 iterations, i.e.

$$\frac{1}{8},\quad \frac{7}{16},\quad \frac{63}{64},\quad \frac{3}{64},\quad \frac{11}{64},\quad \frac{9}{16},\quad \frac{63}{64}.$$

Problem 18. Find the solution of the system

$$\begin{aligned} 5x &= 2 \mod 3\\ 4x &= 7 \mod 9\\ 2x &= 4 \mod 10. \end{aligned}$$

Solution 18. We have $x = 22 \bmod 45$ since

$$\begin{aligned} 5 \cdot 22 &= 110 \Rightarrow 110 - 36 \cdot 3 = 2\\ 4 \cdot 22 &= 88 \Rightarrow 88 - 9 \cdot 9 = 7\\ 2 \cdot 22 &= 44 \Rightarrow 44 - 4 \cdot 10 = 4. \end{aligned}$$

Problem 19. The *Bernoulli numbers* B_0, B_1, B_2, ... are defined by the series

$$\frac{x}{e^x - 1} = \sum_{j=0}^{\infty} \frac{B_j x^j}{j!}$$

where $B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, $B_3 = 0$, $B_4 = -1/30$. Note that $B_{2j+1} = 0$ for all $j = 1, 2, \ldots$. Give an efficient way to calculate the Bernoulli numbers. Then write a C++ program using the `Verylong` and `Rational` class of SymbolicC++.

Solution 19.

Problem 20. In the following arithmetic problem with one multiplication and one sum each digit (except one with the digit 0) has been replaced by an asterisks.

```
  * * * x
    * *
-------
* * * *
```

```
 * * * 0 +
 -------
 * * * *
```

Solve the problem when the asterisks can only be one of the prime numbers 2, 3, 5, 7.

Solution 20. The solution is

```
    7 7 5 x
      3 3
 --------
  2 3 2 5
2 3 2 5 0 +
---------
2 5 5 7 5
```

Problem 21. Consider the nine numbers

$$111,\ 222,\ 333,\ 444,\ 555,\ 666,\ 777,\ 888,\ 999.$$

Find all the postive integers larger 1 that divide these numbers without remainder.

Solution 21. We find 3, 37 and $111 = 3 \cdot 37$.

Problem 22. Let n be a positive integer. Any partition of n into parts $n = p_1 + p_2 + \cdots + p_n$, $0 \le p_1 \le p_2 \le \cdots \le p_n \le n$ can be rearranged in $2 \times r$ matrix

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_r \\ b_1 & b_2 & \cdots & b_r \end{pmatrix}$$

called the *Frobenius symbol*, such that

$$0 \le a_1 < a_2 < \cdots < a_r, \quad 0 \le b_1 < b_2 < \cdots < b_r, \quad n = r + \sum_{j=1}^{r} a_j + \sum_{j=1}^{r} b_j.$$

Each partition has a unique representation as a Frobenius symbol.
(i) Find the Frobenius symbol for $n = 20$.
(ii) Write a C++ program that finds the Frobenius symbol for a given n.

Solution 22. (i) For $n = 20$ we have

$$20 = 5 + 5 + 3 + 3 + 2 + 1 + 1.$$

This can be represented as

```
x   x   x   x   x
x   x   x   x   x
x   x   x
x   x   x
x   x
x
x
```

The diagonal has $r = 3$ elements we remove. Then

```
    x   x   x   x   a_1
x       x   x   x   a_2
x   x               a_3
x   x   x
x   x
x
x
b_1 b_2 b_3
```

The rows on the right are the elements a_1, a_2, a_3 and the columns on the left are the elements b_1, b_2, b_3. Thus we obtain the matrix

$$\begin{pmatrix} 4 & 3 & 0 \\ 6 & 3 & 1 \end{pmatrix}.$$

For each given n one can construct $p(n)$ Frobenius symbols with

$$\sum_{n=0}^{\infty} p(n)q^n = \prod_{n=1}^{\infty} \frac{1}{1-q^n}.$$

Problem 23. Show that (continued-fraction representation)

$$\frac{1}{\sqrt{3}} = [1, 2, 1, 2, 1, 2 \ldots].$$

Solution 23.

Problem 24. The Möbius arithmetic functions $\mu : \mathbb{N} \mapsto \{0, \pm 1\}$ is defined by

$$\mu(n) := \begin{cases} 1 & \text{if} \quad n = 1 \\ (-1)^m & \text{if} \quad n \text{ is the product of } m \text{ distinct primes} \\ 0 & \text{otherwise} \end{cases}$$

Give a C++ implementation using `Verylong` of SymbolicC++ and `BigInteger` using Java. For $m \in \mathbb{N}$ one has

$$\sum_{n|m} \mu(n) = \delta_{m,1}$$

where δ denotes the Kronecker delta.

Solution 24.

$$=$$

Problem 25. Let x, y be integers. Assume they satisfy the equation

$$x^2 - 8y^2 = 17.$$

(i) Show that $(x_0, y_0) = (5, 1)$ is a solution. Show that $(x_0, y_0) = (7, 2)$ is a solution.
(ii) Let $n = 0, 1, 2, \ldots$. Show that if (x_n, y_n) satisfy the equation, then

$$x_{n+1} = 3x_n + 8y_n, \qquad y_{n+1} = x_n + 3y_n$$

also satisfy the equation.

Solution 25.
(ii) We have

$$x_{n+1}^2 - 8y_{n+1}^2 = x_n^2 - 8y_n^2.$$

Problem 26. *Nobles* are irrationals with continued fraction representations of the form

$$\omega = a_0 + 1/(a_1 + 1/(a_2 + \cdots + 1/(1 + 1/(1 + \cdots$$

which eventually have elements all equal to one. Find the number with all a_j's equal to one.

Solution 26.

$$=$$

Problem 27. Let p and q be prime with $p \leq q$. Solve for p and q such that $p + q$ is prime.

Solution 27. Since p and q are prime and $p + q$ is prime, we have $p + q = 2$ (which has no solution for p and q prime) or $p + q$ is odd. Since the sum of two odd numbers is even, one of p and q must be even. Since 2 is the only even prime number and also the smallest prime number $p = 2$. Thus $p + q = q + 2$ must be prime. From q prime and $q + 2$ prime it follows that q must be the first of a pair of twin primes.

Problem 28. Let $n = 0, 1, 2, \ldots$. The *Fermat numbers* F_n are given by

$$F_n = 2^{(2^n)} + 1$$

(i) Show that the Fermat numerbers satisfy the recurrence relation

$$F_{n+1} = (F_n - 1)^2 + 1$$

with $F_0 = 3$.
(ii) Calculate the first ten Fermat numbers using this recurrence relation and the `Verylong` class of SymbolicC++.
(iii) Calculate the first ten Fermat numbers using this recurrence relation and the `BigInteger` class of Java.

Solution 28.

Problem 29. Consider a palindrome consisting of digits. A *palindrome* is a string of characters that reads the same from left to right or from right to left. Show that if $n \in \mathbb{N}$ is a palindrome with an even number of digits, then n is a multiple of 11. For example $3223 : 11 = 293$. Write a C++ program utilizing the string class to find out whether a string with an even number of digits is a palindrome.

Solution 29.

Problem 30. Consider the 6 digit number 142857 with the digits 1, 4, 2, 8, 5, 7. Multiply the number by 2,3,4,5,6. Discuss. What is the connection with 6×6 permutation matrices? Write a C++ program that can do the job.

Solution 30.

Problem 31. Can one find two positive integers m and n such that

$$m \cdot n \cdot (m + n) = 29400.$$

Solution 31. We find $m = 24$, $n = 25$ and owing to symmetry $m = 25$, $n = 24$.

Problem 32. Let $n \in \mathbb{N}$. Show that $7^n - 2^n$ is divisible by 5. Is $7^n - 3^n$ divisible by 4?

Solution 32.

Problem 33. Let $n \in \mathbb{N}$ and $x, y \in \mathbb{R}$. Show that $x^n - y^n$ is divisible by $x - y$.

Solution 33.

Problem 34. We know that there is an positive integer n such that

$$n^5 = 5277319168$$

Find n using that $0^5 = 0$, $1^5 = 1$, $2^5 = ...2$, $3^5 = ...3$, $4^5 = ...4$, $5^5 = ...5$, $6^5 = ...6$, $6^5 = ...6$, $7^5 = ...7$, $8^5 = ...8$, $9^5 = ...9$, i.e. the last digit of the power 5 of the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 is the number themselve.

Solution 34. The number is 88.

Problem 35. Find integer solutions of

$$31n_1 - 16n_2 = 40.$$

Solution 35. We obtain $n_1 = 8$, $n_2 = 16$.

Problem 36. Let $\mathbb{Z}$ be the set of integers. Let $m_1, m_2 \in \mathbb{Z}$. Find all solutions of

$$m_1^2 + m_2^2 = 80.$$

Describe your approach.

Solution 36.

Problem 37. All rational numbers can be represented by terminating continued fractions. For example for 703/29 we have $24 + 7/29$. The reciprocal of $7/29$ is $29/7$. Now $29/7 = 4 + 1/7$. The reciprocal of the fractional part is $1/7$ is 7. The number 7 has no fractional part. Thus

$$\frac{703}{29} = 24 + \frac{1}{4 + 1/7}.$$

Give a C++ implementation of this algorithm utilizing the `Verylong` and `Rational` class of SymbolicC++.

Solution 37.

Problem 38. Let i, j be integers. Give a C++ implementation of the map

$$i \triangleleft j := 2j - i \bmod 6$$

where mod 6 means divide by 6 and keep only the remainder.

Solution 38.

Problem 39. Let $n \in \mathbb{N}$.

(i) Is $2^{3n} - 1$ divisible by 7?
(ii) Is $3^{2n} + 7$ divisible by 8?

=

Problem 40. Let $n \in \mathbb{N}$. Show that $x^n - 1$ is divisible by $x - 1$ for $x \neq 1$.

Chapter 5

Combinatorical Problems

Problem 1. Let $X := \{ x_0 = 0, x_1, x_2, \ldots, x_{n-1} \}$ be a set of n points on the real axis with $x_j < x_{j+1}$ for $j = 0, 1, \ldots, n-2$. Let ΔX denote the sequence of all

$$\binom{n}{2} \equiv \frac{n!}{(n-2)!2!}$$

pairwise distance between points in X with the ordering $\Delta x_j \leq \Delta x_{j+1}$ for $j = 0, 1, \ldots, \binom{n}{2} - 1$. For example, let $X = \{ 0, 2, 4, 7, 10\}$. Then

$$\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}.$$

Given X write a C++ program that generates ΔX. Can one reconstruct X from ΔX? For the given example look at

```
    0 2 4 7 10
 0    2 4 7 10
 2      2 5  8
 4        3  6
 7           3
10
```

Solution 1.

Problem 2. The rank of an element in a sequence (one-dimensional array) of numbers is the number of smaller elements in the sequence plus the number of equal elements that appear to its left. For example, if the sequence is given as the one-dimensional array $a = [4, 3, 9, 3, 7]$, then the ranks are $r = [2, 0, 4, 1, 3]$. Write a C++ program with a function `void rank(T* a,int n,int* r)` that

computes the ranks of the elements of the array $a[0:n-1]$. Once the elements have been ranked using the function `rank()` write a function `rearrange()` that rearrange them in nondecreasing order so that $a[0] \leq a[1] \leq \cdots \leq a[n-1]$ by moving elements to positions corresponding to their ranks.

Solution 2. We can estimate the complexity of the function `rank` by counting the number of comparisons between elements of the array `a`. These comparisons are done in the `if` statement. For each value of `i`, the number of element comparisons is `i`. Thus the total number of element comparisons is $1+2+\cdots+n-1=(n-1)n/2$.

```
// rank.cpp
#include <iostream>
using namespace std;

template<class T>
void rank(T* a,int n,int* r)
{
  for(int i=0;i<n;i++) r[i] = 0;  // initialize
  // compare all elements
  for(int j=1;j<n;j++)
    for(int k=0;k<j;k++)
      if(a[k] <= a[j]) r[j]++;
      else r[k]++;
}

template <class T>
void rearrange(T* a,int n,int* r)
{
  T* u = new T[n+1];
  // move to the correct place in u
  for(int i=0;i<n;i++) u[r[i]] = a[i];
   // move back to array a
   for(int j=0;j<n;j++) a[j] = u[j];
  delete[] u;
}

int main(void)
{
  int n = 5;
  int* a = new int[n];
  a[0] = 4; a[1] = 3; a[2] = 9; a[3] = 3; a[4] = 7;
  int* r = new int[n];
  rank(a,n,r);
  for(int i=0;i<n;i++) cout << "r[" << i << "] = " << r[i] << endl;
  rearrange(a,n,r);
  for(int j=0;j<n;j++) cout << "a[" << j << "] = " << a[j]  << endl;
  delete[] a; delete[] r;
```

```
  return 0;
}
```

Problem 3. Suppose we list all the $2^n - 1$ nonempty subsets of the set of numbers $\{1, 2, \ldots, n\}$. Then, for each subset, we write down the product of its elements. Finally, we add these $2^n - 1$ numbers to obtain the number s_n. Obviously $s_1 = 1$. For $n = 2$ we have $\{1\}$, $\{2\}$, $\{1, 2\}$. Thus $s_2 = 5 = 1 + 2 + 2$. For $n = 3$ the seven products one obtains are 1, 2, 3, 1×2, 1×3, 2×3 and $1 \times 2 \times 3$. Thus $s_3 = 23$.
(i) Find s_4.
(ii) Find a recursion $s_{n+1} = f(n, s_n)$ for s_{n+1} with $s_{=}1$.
(iii) Write a C++ program that implements this recurrence relation with the initial value $s_1 = 1$.

Solution 3. (i) For s_4 we find

$$s_4 = 1 + 2 + 3 + 4 + 2 + 3 + 4 + 6 + 8 + 12 + 6 + 8 + 12 + 24 + 24.$$

Thus $s_4 = 119$.
(ii) We have

$$S_1 = 1, \;\; S_2 = (3 \cdot 1) + 2 = 5, \;\; S_3 = (4 \cdot 5) + 3 = 23, \;\; (5 \cdot 23) + 4 = 119.$$

The recursion is given by

$$s_{n+1} = (n + 2)s_n + (n + 1)$$

with $s_1 = 1$.

Problem 4. How many arrangements of a, a, a, b, b, b, c, c, c are there so that no 2 letters of the same type appear consecutively? For example *abcabcabc* would be such an arrangement. Could a tree structure be used to find the solution? Write a C++ program that finds all sequences.

Solution 4.

Problem 5. A coin is tossed eight times in a row.
(i) What is the probability of getting exactly four heads in a row?
(ii) What is the probability of getting at least four heads in a row?

Solution 5. (i) We partition the set of 8-sequences with exactly 4 heads in a row into five categories:

`HHHHT***, THHHHT**, *THHHHT*, **THHHHT, ***THHHH .`

Each `*` could be either `H` or `T` (wildcard), so the total number is $2^3 + 2^2 + 2^2 + 2^2 + 2^3 = 28$. There are 2^8 sequences of coin tosses since each toss can be one of 2 choices. Therefore, if we assume each sequence of coin tosses is equally likely, the probability of getting 4 heads in a row is $28/256 = 0.109375$.
(ii) The number of ways to obtain exactly 4 heads has been calculated in (i). For exactly 5 heads in a row we have

`HHHHHT**, THHHHHT*, *THHHHHT, **THHHHH`

gives $2^2 + 2 + 2 + 2^2 = 12$ ways. For exactly 6 heads in a row we have

`HHHHHHT*, THHHHHHT, *THHHHHH`

gives $2 + 1 + 2 = 5$ ways. For exactly 7 heads in a row we have

`HHHHHHHT, THHHHHHH`

which gives two ways. For 8 heads `HHHHHHHH` we obviously have one way. This provides a total of $28 + 12 + 5 + 2 + 1 = 48$ ways to obtain at least 4 heads. Therefore, the probability of getting at least 4 heads in a row is $48/256 = 0.1875$.

Problem 6. Show that the number of ways in which n different objects can be arranged in a a ring, if only relative order matters, is $(n-1)!$. First give an example with 3 objects.

Solution 6. For three objects a, b, c we have abc, bac.

Problem 7. (i) How many n-digit ternary sequences (using only 0,1,2) are there with k 1's?
(ii) Find the sequences for $n = 3$ and $k = 2$.
(iii) Write a C++ program that finds these sequences for given n and k in lexicographical order.

Solution 7. Pick the k positions for the 1's in $\binom{n}{k}$ ways. Then for each placement of the 1's there are 2^{n-k} ways to assign 0 or 2 to each of the remaining $n-k$ positions. Thus the total number is

$$\binom{n}{k} \cdot 2^{n-k}.$$

If $n = 3$ and $k = 2$ we have 6 ways. The sequence are in lexicographical order

`011 101 110 112 121 211.`

Problem 8. How many arrangements of $a, a, a, b, b, b, c, c, c$ are there so that no 2 letters of the same type appear consecutively? For example, *abcabcabc* would be such an arrangement.

Solution 8. The total number of arrangements of $a, a, a, b, b, b, c, c, c$ is

$$\frac{9!}{3!3!3!} = 1680$$

which follows from the formula for the permutations of multisets. We use the inclusion-exclusion principle to find the number of arrangements. Let S_a be the multiset of those arrangements that have two a's consecutive. Similarly, we define S_b and S_c. If two a's are consecutive, then glue them to form a block which we consider as one letter. Then $|S_a|$ is the number of ways to permute the multiset $\{aa, b, b, b, c, c, c\}$, which is $7!/(3!3!)$, times the number of distinct ways to insert the last a, namely 7. When inserting the last a we only count the space to the left of aa and not to the right as this would be an over count. Thus

$$|S_a| = 7 \cdot \frac{7!}{3!3!} = 980.$$

Similarly, $|S_b| = |S_c| = S_a|$ by symmetry. Next we construct $S_a \cap S_b$ by finding all multiset permutations on $\{aa, bb, c, c, c\}$ and then inserting an a and a b. There are 5 distinct ways to insert the extra a into the 5-perm, 6 distinct ways to insert the extra b into the 6-perm. In addition, it is possible that a is inserted just to the right of aa and then it is separated by the b on the second step. This is equivalent in finding a multiset permutation of $\{aaba, bb, c, c, c\}$. Therefore

$$|S_a \cap S_b| = \binom{5}{1,1,3} \cdot 5 \cdot 6 + \binom{5}{1,1,3} = 620.$$

Similarly, $|S_c \cap S_b| = |S_a \cap S_c| = 620$. Finally, we calculate the number of elments in the multiset $S_a \cap S_b \cap S_c$, i.e. the number of distinct multiset permutations of $\{aa, a, bb, b, cc, c\}$. There are 6 permutations of $\{aa, bb, cc\}$. We can add the extra a in 4 ways, either to the right of aa or in one of the 3 other positions. In the first case, after inserting the b and c we must end up with one of the strings $aaba$ (with 3 places to put the c) $aaca$ (with three places to put the b) $aabca$ or $aacba$, totally 8 possibilities. In the second case, if the b goes to the right of the bb the c is forced, if the b goes in any of the remaining 4 positions, the c can be inserted anywhere except to the right of cc, giving $3 + 3 \cdot 4 \cdot 5 = 63$ possibilities. Therefore

$$|S_a \cap S_b \cap S_c| = 6 \cdot (8 + 63) = 426.$$

Thus using the inclusion-exclusion principle the number of mutiset permutations with no double letters is $1680 - 3 \cdot 980 + 3 \cdot 620 - 426 = 174$.

Problem 9. Let n be a positive integer. Use the inclusion-exclusion principle to prove that

$$n = \sum_{k=1}^{n} (-1)^{k-1} k \binom{n}{k} 2^{n-k}.$$

Solution 9. Using the identity

$$k\binom{n}{k} \equiv n\binom{n-1}{k-1}$$

we obtain the equivalent identity

$$n = \sum_{k=1}^{n} (-1)^{k-1} n \binom{n-1}{k-1} 2^{n-k}$$

to be proved. Substituting the variables $j = k-1$ and $m = n-1$, we arrive at the equivalent identity

$$1 = \sum_{j=0}^{m} (-1)^j \binom{m}{j} 2^{m-j}.$$

Let S be the set of all binary sequences of length m. Define A_i to be the set of all the binary sequences of length m, which have a 1 in the i-th position. Then, for any j, $1 \le j \le m$, and for any $1 \le i_1 < i_2 < \cdots < i_j \le m$, we have

$$|A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_j}| = 2^{m-j}.$$

Therefore, by inclusion-exclusion, we obtain that the number of all-zero binary sequences of length m (which is 1) is equal to

$$|S - (A_1 \cup A_2 \cup \cdots \cup A_m)| = 2^m - \sum_{j=1}^{m} (-1)^j \binom{m}{j} 2^{m-j} = \sum_{j=0}^{m} (-1)^j \binom{m}{j} 2^{m-j}.$$

Problem 10. Show that the number of ways of writing the positive integer n as a sum of positive integers, where the order of the summands is significant, is 2^{n-1} for $n \ge 1$. For example, for $n = 3$ we have $3 = 3$, $3 = 2 + 1$, $3 = 1 + 2$, $3 = 1 + 1 + 1$.

Solution 10. There is one expression consisting of the single number n. Any other expression ends with some positive integer $j < n$, preceded by an expression with sum $n - j$. Thus the number of expressions satisfies

$$s_n = 1 + \sum_{j=1}^{n-1} s_{n-j} = 1 + \sum_{j=1}^{n-1} s_j.$$

We find that

$$s_{n+1} = s_n + s_n = 2s_n.$$

Since $s_1 = 1$ we have $s_n = 2^{n-1}$.

Problem 11. A derangement (or fixed-point-free permutation) of $\{1, 2, \ldots, n\}$ is a permutation f such that $f(j) \neq j$ for all $j = 1, 2, \ldots, n$. What is the number of derangements of n objects?

Solution 11. There are $n!$ permutations. Using

$$\binom{n}{n} = 1, \qquad \binom{n}{1} = (n-1)!$$

we find that the number of derangements of n objects is given by

$$\begin{aligned} n! - n(n-1)! + \binom{n}{2}(n-2)! - + \cdots + (-1)^n &= n!\left(\frac{1}{2!} - \frac{1}{3!} + - \cdots + (-1)^n \frac{1}{n!}\right) \\ &= n! \sum_{j=2}^{n} (-1)^j \frac{1}{j!}. \end{aligned}$$

Problem 12. A numerical partition of a positive integer n is a sequence

$$p_1 \geq p_2 \geq \cdots \geq p_k \geq 1$$

such that

$$p_1 + p_2 + \cdots + p_k = n.$$

Each p_j is called a part. For example, $18 = 7+4+4+1+1+1$ is a partition of 18 into 6 parts. The number of partitions of n into k parts is denoted by $p(n, k)$.
(i) Find $p(7, 3)$.
(ii) Show that the recurrence for $p(n, k)$ is given by

$$p(n, k) = p(n-1, k-1) + p(n-k, k)$$

with the initial conditions $p(n, 0) = 0$, $p(k, k) = 1$. Obviously, we have $p(n, 1) = 1$.
(iii) Write a C++ program that implements this recurrence..
(iv) Every numerical partition of a positive integer n corresponds to a unique *Ferrer's diagram*. A Ferrer's diagram of a partition is an arrangement of n dots on a square grid, where a part j in the partition is represented by placing p_j dots in a row. This means we represent each term of the partition by a row of dots, the terms in descending order with the largest at the top. Sometimes it is more convenient to use squares instead of dots (in this case the diagram is called a Young diagram). Draw the Ferrer's diagram for the partition $18 = 7+4+4+1+1+1$. The partition we obtain by reading the Ferrer's diagram by columns instead of rows is called the conjugate of the original partition. Find the conjugate of the partition $18 = 7 + 4 + 4 + 1 + 1 + 1$.

Solution 12. (i) We have

$$7 = 5 + 1 + 1 = 4 + 2 + 1 = 3 + 3 + 1 = 3 + 2 + 2.$$

Thus $p(7, 3) = 4$.
(ii) Consider the cases $k = 1$ and $k > 1$. For $k = 1$ we have $p(n, 1) = 1$. For $k > 1$ we use induction.
(iv) We have $18 = 6 + 3 + 3 + 3 + 1 + 1 + 1$.

Problem 13. The *Bell numbers* count (starting from 0) the ways that n distinguishable objects can be grouped into sets if no set can be empty. Thus the Bell numbers are given by the sequence

$$\{\, 1, 1, 2, 5, 15, 52, 203, 877, 4140, \dots \}.$$

For example the numbers 1, 2, 3 can be grouped into sets so that

1) 1, 2, 3 are in three separate sets: $\{1\}$, $\{2\}$, $\{3\}$
2) 1 and 2 are together and 3 is separate: $\{1, 2\}$, $\{3\}$
3) 1 and 3 together and 2 separate: $\{1, 3\}$, $\{2\}$
4) 2 and 3 together and 1 separate: $\{2, 3\}$, $\{1\}$
5) 1, 2, 3 are all together in a single set: $\{1, 2, 3\}$

Hence for $n = 3$ there are five partitions and thus the third Bell number is 5.
(i) Let P_n denote the n^{th} Bell number, i.e. the number of all partitions of n objects. Then we have

$$P_n = \frac{1}{e} \sum_{k=0}^{\infty} \frac{k^n}{k!}.$$

(i) Find a recurrence relation for P_n.
(ii) Write a C++ program that implements this recursion. Apply `Verylong` of SymbolicC++.
(iii) Find the MacLaurin expansion (expansion around 0) of $\exp(\exp(x))$ and establish a connection with the Bell numbers.

Solution 13. (i) Let S be the set to be partioned and $x \in S$. If the class containing x has k elements, it can be chosen in

$$\binom{n-1}{k-1}$$

ways. The remaining $n - k$ elements can be partioned in P_{n-k} ways. Thus the number of partitions in which the class containing x has k elements is

$$\binom{n-1}{k-1} P_{n-k}.$$

This remains true for $k = n$ if we set $P_0 = 1$. Thus

$$P_n = \sum_{k=1}^{n} \binom{n-1}{k-1} P_{n-k} = \sum_{k=0}^{n-1} \binom{n-1}{k} P_k.$$

(iii) Since $d(\exp(\exp(x))/dx = \exp(x)\exp(\exp(x))$ etc and $\exp(0) = 1$ we find

$$\exp(\exp(x)) = e\left(1 + \frac{1x}{1!} + \frac{2x^2}{2!} + \frac{5x^3}{3!} + \cdots\right)$$

Thus the coefficients 1, 1, 2, 5, ... are the Bell numbers.

Problem 14. Let N be a positive integer. Consider the set of numbers

$$S = \{\, 0,\, 1,\, 2, \ldots, N \,\}$$

How many pairs (m, n) $(m, n \in S)$ can be formed with the condition that $m < n$.

Solution 14. First we can form

$$(0, 1),\, (0.2),\, \ldots, (0, N)$$

which are N pairs. Then we can form

$$(1, 2),\, (1, 3),\, \ldots, (1, N)$$

which are $N - 1$ pairs. Proceeding in this way we obtain

$$N + (N-1) + (N_2) + \cdots + 2 + 1 \equiv \frac{N(N+1)}{2}$$

ways.

Problem 15. Consider a bitstring of length m which has exactly m_1 ones and m_2 zeros $(m_1 + m_2 = m)$.
(i) Find the number of different possible bitstrings.
(ii) Consider the special case $m = 4$, $m_1 = m_2 = 2$. Write down the bitstrings in lexicographical order.

Solution 15. We have

$$\binom{m}{m_1} \equiv \frac{m!}{m_1! m_2!}.$$

(ii) For the special case we find

$$\binom{4}{2} = \frac{4!}{2!2!} = 6$$

with

$$0011, \quad 0101, \quad 0110, \quad 1001, \quad 1010, \quad 1100.$$

Problem 16. A dice is thrown twice. The first throw determines the tens digit and the second throw the ones digit of the two-digit number. Find the probability that this two-digit number is a perfect square.

Solution 16. Since the dice is thrown twice there are 36 possible outcomes, namely

$$11, 12, \ldots 16, 21, \ldots, 26, \quad \ldots \quad , 61, \ldots, 66.$$

Only four of them represent squares, namely

$$16, \quad 25, \quad 36, \quad 64.$$

Thus the probability is $4/36 = 1/9$.

Problem 17. How many different numbers of 7 digits can be formed with the digits 1122334 ?

Solution 17.

$$=$$

Problem 18. Suppose three fair coins are flipped. Let X be the event that they same face. Let Y be the event that there is at most one head. Show that X and Y are independent.

Solution 18. There are eight possible outcomes. We have

$$|X| = |\{TTT, HHH\}| = 2, \quad |Y| = |\{TTT, HTT, THT, TTH\}| = 4.$$

Consequently

$$|X \cap Y| = |\{TTT\}| = 1.$$

It follows that

$$P(X \cup Y) = P(TTT) = \frac{1}{8}, \quad P(X) = P(TTT \cup HHH) = \frac{1}{4}, \quad P(Y) = \frac{1}{2}.$$

Thus $P(X \cap Y) = P(X)P(Y)$.

Problem 19. The *Stirling number* of the second kind $S(n, k)$ is the number of partitions of a set with n elements into k classes. Let $b^\dagger$, b be Bose creation and annihilation operators with the commutation relations

$$[b, b^\dagger] = bb^\dagger - b^\dagger b = I$$

where I is the identity operator. Then $S(n,k)$ can be defined by

$$(b^\dagger b)^n = \sum_{k=1}^{n} S(n,k)(b^\dagger)^k b^k.$$

Let $n = 3$. Use this definition to find $S(3,1)$, $S(3,2)$, $S(3,3)$.

Solution 19. Using the commutation relation we find

$$(b^\dagger b)^3 = b^\dagger b + 3b^\dagger b^\dagger bb + b^\dagger b^\dagger b^\dagger bbb.$$

Thus

$$S(3,1) = 1, \quad S(3,2) = 3, \quad S(3,3) = 1.$$

Problem 20. Let $n \geq 1$. Let a_1, a_2, ...,a_n be real numbers. How many terms are there in the sum

$$\sum \sum_{1 \leq j_1 < j_2 < j_3 \leq n} \sum a_{j_1} a_{j_2} a_{j_3}.$$

Solution 20.

=

Problem 21. Let $n \geq 1$. Let $c_1^\dagger$, $c_2^\dagger$, ..., $c_n^\dagger$ be spin-less Fermi creation operators and c_1, c_2, ..., c_n be spin-less Fermi annihilation operators. Thus

$$[c_j^\dagger, c_k]_+ = \delta_{jk} I, \quad [c_j, c_k]_+ = 0, \quad [c_j^\dagger, c_k^\dagger]_+ = 0$$

where 0 is the zero operator, $[\,,\,]_+$ denotes the anti-commutator and $j, k = 1, \ldots, n$. Then $(c_j)^2 = 0$ and $(c_j^\dagger)^2 = 0$. Let $|0\rangle$ be the vacuum state with $c_j|0\rangle = 0|0\rangle$ and $j = 1, \ldots, n$. Then for $n = 1$ we can form the two-dimensional basis $|0\rangle$, $c_1^\dagger|0\rangle$. For $n = 2$ we can form the four dimensional basis

$$|0\rangle, \quad c_1^\dagger|0\rangle, \quad c_2^\dagger|0\rangle, \quad c_2^\dagger c_1^\dagger|0\rangle.$$

(i) Find the basis for $n = 3$.
(ii) Find the basis for $n = 4$.
(iii) Extend to arbitrary n.

Solution 21. (i) For $n = 3$ we have one state $|0\rangle$ with no Fermi particle, three states with one Fermi particle

$$c_1^\dagger|0\rangle, \quad c_2^\dagger|0\rangle, \quad c_3^\dagger|0\rangle,$$

three states with two Fermi particle

$$c_1^\dagger c_2^\dagger|0\rangle, \quad c_1^\dagger c_3^\dagger|0\rangle, \quad c_2^\dagger c_3^\dagger|0\rangle$$

and one state with three Fermi particles $c_1^\dagger c_2^\dagger c_3^\dagger|0\rangle$. Altogether we have $2^3 = 8$ states.
(ii) For $n = 4$ we have 16 states

$$|0\rangle, \quad c_1^\dagger|0\rangle, \quad c_2^\dagger|0\rangle, \quad c_3^\dagger|0\rangle, \quad c_4^\dagger|0\rangle,$$

$$c_1^\dagger c_2^\dagger|0\rangle, \quad c_1^\dagger c_3^\dagger|0\rangle, \quad c_1^\dagger c_4^\dagger|0\rangle, \quad c_2^\dagger c_3^\dagger|0\rangle, \quad c_2^\dagger c_4^\dagger|0\rangle, \quad c_3^\dagger c_4^\dagger|0\rangle$$

$$c_1^\dagger c_2^\dagger c_3^\dagger|0\rangle, \quad c_1^\dagger c_2^\dagger c_4^\dagger|0\rangle, \quad c_1^\dagger c_3^\dagger c_4^\dagger|0\rangle, \quad c_2^\dagger c_3^\dagger c_4^\dagger|0\rangle, \quad c_1^\dagger c_2^\dagger c_3^\dagger c_4^\dagger|0\rangle.$$

(iii) For general n we have 2^n states. Note that

$$2^n \equiv \sum_{k=0}^{n} \binom{n}{k}.$$

First we have one vacuum state $|0\rangle$, then $n \equiv \binom{n}{1}$ one Fermi particle states, $\binom{n}{2}$ states with two Fermi particles, $\binom{n}{3}$ states with three Fermi particles etc and finally the state with n Fermi particle $c_n^\dagger c_{n-1}^\dagger \cdots c_1^\dagger|0\rangle$.

Problem 22. In how many ways can four people arrange themselves in a row? (open end boundary conditions) In how many ways can four people arrange themselves around a circular table? (cyclic boundary conditions, circular permutation) Number the people as 0, 1, 2, 3. Give all configuration for the second case. Draw pictures for all possible configurations starting with

```
0 -- 1
|    |
3 -- 2
```

Solution 22.

Problem 23. Given eight soccer teams we number 0, 1, ..., 7. Every team plays every otherteam. At each weekend there are four matches with each team playing. To generate all the matches for the seven weekends we proceed as follows: For the first weekend the matches are

```
(0,7)  (1,6)  (2,5)  (4,5)
```

We keep the 7 in the first pairing fixed for all weekends. For all other entries we add -1 modulo 7. So the pairings for the second weekend are

```
(6,7)  (0,5)  (1,4)  (2,3)
```

Give a C++ implementation to find the pairings for all weekends.

Solution 23. The C++ code is

```
// soccer.cpp
// c++ -std=c++11 -o soccer soccer.cpp
#include <iostream>
using namespace std;

int main(void)
{
int teams = 8;
int i, j, team1, team2;

for(i=0;i<teams-1;++i)
{
for(j=0;j<teams/2;++j)
{
 team1 = (2*teams-2-j-i) % (teams-1);
 team2 = (2*teams-2+j-i) % (teams-1);
 if(j==0) team2 = teams-1;
 cout << "(" << team1 << "," << team2 << ") ";
}
cout << endl;
}
}
```

The output is

```
 (0,7)  (1,6)  (2,5)  (3,4)
 (6,7)  (0,5)  (1,4)  (2,3)
 (5,7)  (6,4)  (0,3)  (1,3)
 (4,7)  (5,3)  (6,2)  (0,2)
 (3,7)  (4,2)  (5,1)  (6,0)
 (2,7)  (3,1)  (4,0)  (5,6)
 (1,7)  (2,0)  (3,6)  (4,5)
```

Obviously the code could also be used for 16 teams by setting `int teams=16;`.

Problem 24. The English language has 26 letters. A palindrome is a word that reads the same forwards and backwards. How many five letter, one-word, English language palindromes are possible? Describe how you find your solution.

Solution 24.

Chapter 6

Matrix Calculus

Problem 1. An $n \times n$ matrix T is called a *Toeplitz matrix* if it satisfies the relation $T(k, j) = T(k-1, j-1)$, for $2 \leq j, k \leq n$. In other words, the entries on each diagonal of T are all equal. Hence, such a matrix is determined by the $2n-1$ entries appearing in the first row and first column. We denote these entries by $t_0, t_1, \ldots, t_{2n-2}$ such that

$$T = \begin{pmatrix} t_{n-1} & t_{n-2} & \cdots & & t_2 & t_1 & t_0 \\ t_n & t_{n-1} & t_{n-2} & \cdots & & t_2 & t_1 \\ t_{n+1} & t_n & t_{n-1} & t_{n-2} & \cdots & & t_2 \\ \vdots & & & \ddots & & & \vdots \\ \vdots & & & & \ddots & & \vdots \\ t_{2n-3} & t_{2n-2} & \cdots & & & t_{n-1} & t_{n-2} \\ t_{2n-2} & t_{2n-3} & \cdots & & t_{n+1} & t_n & t_{n-1} \end{pmatrix}.$$

We say that the vector $\mathbf{t} = (t_0, t_1, \ldots, t_{2n-2})$ defines the Toeplitz matrix T. Thus the 4×4 Toeplitz matrix defined by the vector $\mathbf{t} = (t_0, t_1, t_2, t_3, t_4, t_5, t_6)$ is

$$T = \begin{pmatrix} t_3 & t_2 & t_1 & t_0 \\ t_4 & t_3 & t_2 & t_1 \\ t_5 & t_4 & t_3 & t_2 \\ t_6 & t_5 & t_4 & t_3 \end{pmatrix}.$$

Write a C++ program that generates the Toeplitz matrix T from a given vector $\mathbf{t}$. Vice versa given a Toeplitz matrix T find the vector $\mathbf{t}$.

Solution 1.

```
// toeplitz.cpp
```

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

template <class T>
vector<vector<T> > toeplitz(const vector<T> &t)
{
 int n = (t.size()+1)/2;
 vector<vector<T> > toepl(n);

 for(int i=0;i<n;i++)
 {
  toepl[i].resize(n);
  for(int j=0;j<n;j++) toepl[i][j] = t[n-1-j+i];
 }
 return toepl;
}

template <class T>
vector<T> toepl_vector(const vector<vector<T> > &toepl)
{
 int i, n = toepl.size();
 vector<T> t(2*toepl.size()-1);
 for(i=0;i<n;i++) t[i] = toepl[0][n-1-i];
 for(i=1;i<n;i++) t[i+n-1] = toepl[n-1][n-1-i];
 return t;
}

int main(void)
{
 int i, j;
 vector<string> v(7);
 v[0] = "t0"; v[1] = "t1"; v[2] = "t2"; v[3] = "t3";
 v[4] = "t4"; v[5] = "t5"; v[6] = "t6";

 vector<vector<string> > tp = toeplitz(v);
 vector<string> t = toepl_vector(tp);

 for(i=0;i<4;i++)
 {
  cout << "[ ";
  for(j=0;j<4;j++) cout << tp[i][j] << " ";
  cout << "]" << endl;
 }
 cout << endl;
 for(i=0;i<7;i++) cout << "[ " << t[i] << " ]" << endl;
```

```
 return 0;
}
```

Problem 2. Let $A_1, A_2, \ldots, A_p$ be square matrices of the same size. Let

$$f_{n,1}(A_1, A_2, \ldots, A_p) := (e^{A_1/n} e^{A_2/n} \cdots e^{A_p/n})^n.$$

Then

$$\| \exp(\sum_{j=1}^{p} A_j) - f_{n,1}(A_1, A_2, \ldots, A_p)\| \leq \frac{2}{n} \left(\sum_{j=1}^{p} \|A_j\| \right)^2 \exp\left(\frac{n+2}{n} \sum_{j=1}^{p} \|A_j\| \right)$$

and

$$\lim_{n\to\infty} f_{n,1}(A_1, A_2, \ldots, A_p) = \exp\left(\sum_{j=1}^{p} A_j \right).$$

For $p = 2$ we obtain

$$e^{A_1+A_2} = \lim_{n\to\infty} (e^{A_1/n} e^{A_2/n})^n. \tag{1}$$

Let

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \qquad A_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \qquad A_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Then $A = A_1 + A_2$. Note that $[A_1, A_2] \neq 0$.
(i) Calculate e^A by diagonalizing A.
(ii) Calculate e^A using equation (1).
(iii) Calculate $f_{2,1}(A_1, A_2)$ and the error estimation.

Solution 2.

Problem 3. The discrete Fourier transform over n points can be written in matrix form

$$F_n := \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & w & w^2 & \ldots & w^{n-1} \\ 1 & w^2 & w^4 & \ldots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \ldots & w^{(n-1)^2} \end{pmatrix}$$

where $w := e^{2\pi i/n}$ is the n-th root of unity. We obtain the discrete Fourier transform from

$$(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)^T = F_n (x_1, x_2, \ldots, x_n)^T.$$

Apply F_4 to the data $(1, 0, 0, 1)^T$ and $(0, 1, 1, 0)^T$ and interpret the results to find the underlying periodicity.

Solution 3. We have $w = e^{2\pi i/4} = e^{\pi i/2}$ and

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{\pi i/2} & e^{2\pi i/2} & e^{3\pi i/2} \\ 1 & e^{2\pi i/2} & e^{4\pi i/2} & e^{6\pi i/2} \\ 1 & e^{3\pi i/2} & e^{6\pi i/2} & e^{9\pi i/2} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}.$$

Consequently for the first case we find

$$\begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1-i \\ 0 \\ 1+i \end{pmatrix}$$

and for the second case

$$\begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ -1+i \\ 0 \\ -1-i \end{pmatrix}.$$

The nonzero $\hat{x}_k$ identify frequencies, thus n/k yields the periodicity. In both cases we find periodicity $4/1 = 4$ and $4/3$ (which does not make sense). This result is due to uncertainty about the truncated sequence, i.e. we could have

$$1001100110011001\ldots$$

or

$$1001001001001001\ldots$$

The first sequence has period 4 while the second has period 3.

Problem 4. (i) The discrete Fourier transform over n points can be written in matrix form

$$F_n := \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & w & w^2 & \ldots & w^{n-1} \\ 1 & w^2 & w^4 & \ldots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \ldots & w^{(n-1)^2} \end{pmatrix}$$

where $w = e^{2\pi i/n}$ is the n-th root of unity. Show that the matrix $\frac{1}{\sqrt{n}} F_n$ is unitary.

(ii) Use trigonometric interpolation to find

$$f(x) = c_0 + c_1 e^{ix} + c_2 e^{2ix} + c_3 e^{3ix}$$

which interpolates the points

$$x_0 = 0, \quad y_0 = 0, \qquad x_1 = \frac{\pi}{2}, \quad y_1 = 1,$$

$$x_2 = \pi, \quad y_2 = 1, \qquad x_3 = \frac{3\pi}{2}, \quad y_3 = 0$$

i.e. solve the equation $F_4(c_0, c_1, c_2, c_3)^T = (y_0, y_1, y_2, y_3)^T$.

Solution 4. (i) We have

$$\begin{aligned}\left(\frac{1}{\sqrt{n}} F_n \frac{1}{\sqrt{n}} F_n^*\right)_{j,k} &= \frac{1}{n} \sum_{l=0}^{n-1} w^{jl} (w^*)^{kl} \\ &= \frac{1}{n} \sum_{l=0}^{n-1} e^{(j-k)2l\pi i/n} \\ &= \begin{cases} 1 & j = k \\ \frac{1 - e^{n(j-k)2l\pi i/n}}{1 - e^{(j-k)2l\pi i/n}} & j \neq k \end{cases} \\ &= \delta_{jk}\end{aligned}$$

and

$$\begin{aligned}\left(\frac{1}{\sqrt{n}} F_n^* \frac{1}{\sqrt{n}} F_n\right)_{j,k} &= \frac{1}{n} \sum_{l=0}^{n-1} w^{kl} (w^*)^{jl} \\ &= \frac{1}{n} \sum_{l=0}^{n-1} e^{(k-j)2l\pi i/n} \\ &= \begin{cases} 1 & j = k \\ \frac{1 - e^{n(k-j)2l\pi i/n}}{1 - e^{(k-j)2l\pi i/n}} & j \neq k \end{cases} \\ &= \delta_{kj}.\end{aligned}$$

Consequently $\frac{1}{\sqrt{n}} F_n \frac{1}{\sqrt{n}} F_n^* = \frac{1}{\sqrt{n}} F_n^* \frac{1}{\sqrt{n}} F_n = I$.

(ii) Obviously $(c_0, c_1, c_2, c_3)^T = F_4^{-1} (y_0, y_1, y_2, y_3)^T$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & +i & -1 & -i \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{-1-i}{4} \\ 0 \\ \frac{-1+i}{4} \end{pmatrix}$$

so that

$$f(x) = \frac{1}{2} - \frac{1}{4}(1+i)e^{ix} - \frac{1}{4}(1-i)e^{3ix} = \frac{1}{2} - \frac{1}{2}e^{2ix}(\cos x + \sin x).$$

Problem 5. A *1-inverse* of the $m \times n$ matrix A is an $n \times m$ matrix A^- such that $AA^-A = A$.

(i) Suppose that $m = n$ and that A^{-1} exists, find A^-.
(ii) Is the 1-inverse unique? Prove or disprove.
(iii) The *Moore-Penrose pseudoinverse* of the $m \times n$ matrix A is the 1-inverse A^- of A which additionally satisfies

$$A^-AA^- = A^- \qquad (AA^-)^* = AA^- \qquad (A^-A)^* = A^-A.$$

Let $A = U\Sigma V^*$ be the singular value decomposition of A. Show that $A^- = V\Sigma^-U^*$ is a Moore-Penrose pseudoinverse of A, where

$$(\Sigma^-)_{jk} = \begin{cases} \frac{1}{(\Sigma)_{kj}} & (\Sigma)_{kj} \neq 0 \\ 0 & (\Sigma)_{kj} = 0 \end{cases}$$

Hint: First show that Σ^- is the Moore-Penrose pseudoinverse of Σ. Find the Moore-Penrose pseudoinverse of

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Solution 5. (i) Since A^{-1} exists we find

$$A^{-1}(AA^-A)A^{-1} = A^{-1}AA^{-1}$$

i.e. $A^- = A^{-1}$.
(ii) Consider

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

then both A and the 2×2 identity matrix I_2 are 1-inverses of A. Consequently, the 1-inverse is in general not unique.
(iii) First we calculate ($j, k \in \{1, 2, \ldots, n\}$)

$$(\Sigma^-\Sigma)_{jk} = \sum_{l=1}^{m} (\Sigma^-)_{jl}(\Sigma)_{lk} = \sum_{\substack{l=1 \\ (\Sigma)_{lj} \neq 0}}^{m} \frac{(\Sigma)_{lk}}{(\Sigma)_{lj}}.$$

Since $(\Sigma)_{lj} = 0$ when $l \neq j$ we find

$$(\Sigma^-\Sigma)_{jk} = \begin{pmatrix} \frac{(\Sigma)_{jk}}{(\Sigma)_{jj}} & (\Sigma)_{jj} \neq 0 \\ 0 & (\Sigma)_{jj} = 0 \end{pmatrix} = \begin{cases} \delta_{jk} & (\Sigma)_{jj} \neq 0 \\ 0 & (\Sigma)_{jj} = 0 \end{cases}.$$

Similarly ($j, k \in \{1, 2, \ldots, m\}$)

$$(\Sigma\Sigma^-)_{jk} = \begin{cases} \delta_{jk} & (\Sigma)_{jj} \neq 0 \\ 0 & (\Sigma)_{jj} = 0 \end{cases}.$$

The matrix $\Sigma^-\Sigma$ is a diagonal $n \times n$ matrix while $\Sigma\Sigma^-$ is a diagonal $m \times m$ matrix. All the entries of these matrices are 0 or 1 so that

$$(\Sigma\Sigma^-)^* = \Sigma\Sigma^-, \qquad (\Sigma^-\Sigma)^* = \Sigma^-\Sigma.$$

Now

$$\begin{aligned}(\Sigma\Sigma^-\Sigma)_{jk} &= \sum_{l=1}^{n}(\Sigma)_{jl}(\Sigma^-\Sigma)_{lk} = (\Sigma)_{jj}(\Sigma^-\Sigma)_{jk} \\ &= \begin{cases} \delta_{jk}(\Sigma)_{jj} & (\Sigma)_{jj} \neq 0 \\ 0 & (\Sigma)_{jj} = 0 \end{cases} \\ &= (\Sigma)_{jk}\end{aligned}$$

and

$$\begin{aligned}(\Sigma^-\Sigma\Sigma^-)_{jk} &= \sum_{l=1}^{m}(\Sigma^-)_{jl}(\Sigma\Sigma^-)_{lk} = (\Sigma^-)_{jj}(\Sigma\Sigma^-)_{jk} \\ &= \begin{cases} \delta_{jk}(\Sigma^-)_{jj} & (\Sigma)_{jj} \neq 0 \\ 0 & (\Sigma)_{jj} = 0 \end{cases} \\ &= (\Sigma^-)_{jk}\end{aligned}$$

i.e.

$$\Sigma\Sigma^-\Sigma = \Sigma, \qquad \Sigma^-\Sigma\Sigma^- = \Sigma^-$$

so that Σ^- is the Moore-Penrose pseudoinverse of Σ. The remaining properties are easy to show

$$AA^-A = (U\Sigma V^*)(V\Sigma^-U^*)(U\Sigma V^*) = U\Sigma\Sigma^-\Sigma V^* = U\Sigma V^* = A,$$

$$A^-AA^- = (V\Sigma^-U^*)(U\Sigma V^*)(V\Sigma^-U^*) = V\Sigma^-\Sigma\Sigma^-U^* = V\Sigma^-U^* = A^-,$$

$$(AA^-)^* = (U\Sigma V^*V\Sigma^-U^*)^* = I_m^* = I_m = AA^-,$$

$$(A^-A)^* = (V\Sigma^-U^*U\Sigma V^*)^* = I_n^* = I_n = AA^-.$$

Thus A^- is a Moore-Penrose pseudoinverse of A.
(iv) We find

$$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}^* \begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix} = \begin{pmatrix}2 & 2\\ 2 & 2\end{pmatrix}$$

with eigenvalues 4 and 0 with corresponding orthonormal eigenvectors $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\end{pmatrix}$ and $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\end{pmatrix}$. Thus the singular value decomposition is

$$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix}\begin{pmatrix}2 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix}.$$

The Moore-Penrose pseudoinverse is

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Problem 6. Find the eigenvectors and generalized eigenvectors of

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}.$$

Solution 6. The eigenvalues are all 0. The eigenvectors follow from solving

$$\left(\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} - 0 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

for x_1, x_2 and x_3. Thus $x_2 = 0$. The set of eigenvectors is given by

$$\left\{ \begin{pmatrix} u \\ 0 \\ v \end{pmatrix} \;\middle|\; (u, v) \in \mathbb{C}^2/(0,0) \right\}.$$

The generalized eigenvectors follow from solving

$$\left(\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} - 0 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right)^3 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

for x_1, x_2 and x_3. This equation is satisfied for all x_1, x_2 and x_3. The set of generalized eigenvectors is given by

$$\mathbb{C}^3/(0,0,0).$$

Problem 7. Let

$$A = \begin{pmatrix} 0 & i\pi & 0 \\ 0 & 0 & 0 \\ 0 & -i\pi & 0 \end{pmatrix}.$$

Calculate $\exp(A)$, $\sinh(A)$, $\cosh(A)$, $\sin(A)$, $\cos(A)$ efficiently.

Solution 7. Since the square of the matrix is the 3×3 zero matrix we find

$$\exp \begin{pmatrix} 0 & i\pi & 0 \\ 0 & 0 & 0 \\ 0 & -i\pi & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & i\pi & 0 \\ 0 & 0 & 0 \\ 0 & -i\pi & 0 \end{pmatrix} = \begin{pmatrix} 1 & i\pi & 0 \\ 0 & 1 & 0 \\ 0 & -i\pi & 1 \end{pmatrix}.$$

Using the Cayley-Hamilton theorem, we need to solve the equations

$$\begin{aligned} e^{\lambda} &= c_0 + c_1\lambda + c_2\lambda^2 \\ e^{\lambda} &= c_1 + 2c_2\lambda \\ e^{\lambda} &= 2c_2 \end{aligned}$$

where $\lambda = 0$. The last two equations we obtain by repeated differentiation of the first equation. Thus $c_2 = \frac{1}{2}$ and $c_0 = c_1 = 1$. Consequently

$$\begin{aligned} \exp\begin{pmatrix} 0 & i\pi & 0 \\ 0 & 0 & 0 \\ 0 & -i\pi & 0 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & i\pi & 0 \\ 0 & 0 & 0 \\ 0 & -i\pi & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & i\pi & 0 \\ 0 & 0 & 0 \\ 0 & -i\pi & 0 \end{pmatrix}^2 \\ &= \begin{pmatrix} 1 & i\pi & 0 \\ 0 & 1 & 0 \\ 0 & -i\pi & 1 \end{pmatrix}. \end{aligned}$$

Problem 8. Solve the system of linear equations

$$\begin{pmatrix} 2 & 1 & 1 \\ 0 & 2 & 1 \\ 1 & 0 & 2 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

using the Jacobi method. Start from the initial guess $x_0 = y_0 = z_0 = 0$.

Solution 8. The determinant of the 3×3 matrix is nonzero. Thus the inverse of the 3×3 matrix exists and is given by

$$\begin{pmatrix} 4/7 & -2/7 & -1/7 \\ 1/7 & 3/7 & -2/7 \\ -2/7 & 1/7 & 4/7 \end{pmatrix}.$$

Thus we find for the solution

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4/7 & -2/7 & -1/7 \\ 1/7 & 3/7 & -2/7 \\ -2/7 & 1/7 & 4/7 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/7 \\ 2/7 \\ 3/7 \end{pmatrix}.$$

Problem 9. How would one store a matrix using a linked list?

Solution 9.

Problem 10. Given an $n \times n$ matrix over $\mathbb{C}$. How would we efficiently test whether the matrix is unitary? Apply your approach to

$$U = \frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1 \\ i & i & -i \\ i & -i & -i \end{pmatrix}.$$

Solution 10. Obviously we would not test whether the matrix is invertible and if so calculate the inverse and then test whether $U^* = U^{-1}$. We would just calculate U^* from U and then test entry by entry whether $U^*U = I_n$.

Problem 11. The sum $1^3 + 2^3 + 3^3 + \cdots + n^3$ can be written as

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = an^4 + bn^3 + cn^2$$

where the unknown coefficients a, b, c can be determined from a system of linear equations obtained from $n = 1$, $n = 2$, $n = 3$. Find this system of linear equations and write a C++ program using *Gauss elimination* that finds the solution.

Solution 11.

Problem 12. Let A be an $n \times n$ symmetric matrix over $\mathbb{R}$. Write a C++ program that uses *Givens transform* to cast the matrix into *tridiagonal form*.

Solution 12. We apply the C++ program to the matrix

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}.$$

We obtain

$$\widetilde{A} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

```
// Givens.cpp

#include <iostream>
#include <cmath>
using namespace std;

void rotation(double** A,int n)
{
  const double pi = 3.14159265;
  int i, j, k;
  for(j=0;j<n-2;j++)  // loop over rows
  {
  for(i=j+2;i<n;i++)  // loop over columns
  {
  double r = sqrt(A[j+1][j]*A[j+1][j]+A[i][j]*A[i][j]);
  if(r < 1E-16) continue;
```

```
  // calculate c = cos(phi), s = sin(phi)
  double c = A[j+1][j]/r; double s = -A[i][j]/r;
  // rotate rows
  for(k=j;k<n;k++)
  {
  double h = A[i][k]*c + A[j+1][k]*s;
  A[j+1][k] = A[i][k]*s - A[j+1][k]*c;
  A[i][k] = h;
  }
  // rotate columns
  for(k=j;k<n;k++)
  {
  double h = A[k][i]*c + A[k][j+1]*s;
  A[k][j+1] = A[k][i]*s - A[k][j+1]*c;
  A[k][i] = h;
  } // end for loop
  }
  }
}

int main(void)
{
 double** A = NULL;
 int n = 4;
 A = new double* [n];
 for(int k=0;k<n;k++)
 A[k] = new double[n];
 A[0][0] = 1.0; A[0][1] = 0.0; A[0][2] = 0.0; A[0][3] = 1.0;
 A[1][0] = 0.0; A[1][1] = 1.0; A[1][2] = 1.0; A[1][3] = 0.0;
 A[2][0] = 0.0; A[2][1] = 1.0; A[2][2] = -1.0; A[2][3] = 0.0;
 A[3][0] = 1.0; A[3][1] = 0.0; A[3][2] = 0.0; A[3][3] = -1.0;
 rotation(A,n);
 for(int i=0;i<n;i++)
 for(int j=0;j<n;j++)
 { cout << "A[" << i << "][" << j << "] = " << A[i][j] << endl; }
 for(int p=0;p<n;p++) delete[] A[p]; delete[] A;
 return 0;
}
```

Problem 13. Given an $n \times n$ tridiagonal matrix over $\mathbb{R}$ with $n \geq 3$. Write a C++ program that finds the characteristic polynomial.

Solution 13.

```
// charPolynomial.cpp

#include <iostream>
#include <cmath>
```

```
using namespace std;

void charPoly(double** A,double* c,int n)
{
  double* c0 = new double[n+1];
  double* c1 = new double[n+1];
  // initializing
  c0[0] = A[n-1][n-1];
  c0[1] = -1.0;
  c1[0] = A[n-2][n-2]*A[n-1][n-1] - A[n-1][n-2]*A[n-2][n-1];
  c1[1] = -(A[n-1][n-1] + A[n-2][n-2]);
  c1[2] = 1.0;

  for(int i=3;i<=n;i++) // recursive loop
  {
  c[i] = -c1[i-1];
  for(int j=i-1;j>=0;j--)
  {
  c[j] = A[n-i][n-i]*c1[j] - A[n-i][n-i+1]*A[n-i+1][n-i]*c0[j];
  }
  for(int k=i-2;k>=0;k--) { c[k+1] -= c1[k]; }
  for(int q=0;q<=i;q++) { c0[q] = c1[q]; c1[q] = c[q]; }
  }
  delete[] c0; delete[] c1;
}

int main(void)
{
  double** A = NULL;
  int n = 4;
  A = new double* [n];
  for(int k=0;k<n;k++)
  A[k] = new double[n];
  A[0][0] = 1.0; A[0][1] = 1.0; A[0][2] = 0.0; A[0][3] = 0.0;
  A[1][0] = 1.0; A[1][1] = -1.0; A[1][2] = 0.0; A[1][3] = 0.0;
  A[2][0] = 0.0; A[2][1] = 0.0; A[2][2] = -1.0; A[2][3] = 1.0;
  A[3][0] = 0.0; A[3][1] = 0.0; A[3][2] = 1.0; A[3][3] = 1.0;
  double* c = new double[n+1];
  charPoly(A,c,n);
  for(int i=0;i<=n;i++)
  { cout << "c[" << i << "] = " << c[i] << endl; }
  for(int p=0;p<n;p++) delete[] A[p]; delete[] A;
  delete[] c;
  return 0;
}
```

Problem 14. Let A be an $n \times n$ matrix over $\mathbb{R}$. Let $\mathbf{u}$ be a nonzero column vector in $\mathbb{R}^n$. Computing the $n \times n$ matrix

$$\left(I_n - \frac{2\mathbf{u}\mathbf{u}^T}{\mathbf{u}^T\mathbf{u}}\right) A$$

can be done as follows.

Step 1. Compute the number $\beta = 2/(\mathbf{u}^T\mathbf{u})$.

Step 2. for $j = 1, 2, \ldots, n$ do
$\alpha = u_1 a_{1j} + u_2 a_{2j} + \cdots + u_n a_{nj}$
$\alpha = \beta \cdot \alpha$
for $i = 1, 2, \ldots, n$ do $a_{ij} = a_{ij} - \alpha u_i$

Write a C++ program that implements this algorithm.

Solution 14.

Problem 15. Consider the two polynomials

$$p_1(x) = a_0 + a_1 x + \cdots + a_n x^n, \qquad p_2(x) = b_0 + b_1 x + \cdots + b_m x^m$$

where $n = \deg(p_1)$ and $m = \deg(p_2)$. Assume that $n > m$. Let $r(x) = p_2(x)/p_1(x)$. We expand $r(x)$ in powers of $1/x$, i.e.

$$r(x) = \frac{c_1}{x} + \frac{c_2}{x^2} + \cdots$$

From the coefficients c_1, c_2, …, c_{2n-1} we can form an $n \times n$ *Hankel matrix*

$$H_n = \begin{pmatrix} c_1 & c_2 & \cdots & c_n \\ c_2 & c_3 & \cdots & c_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & \cdots & c_{2n-1} \end{pmatrix}.$$

The determinant of this matrix is proportional to the *resultant* of the two polynomials. If the resultant vanishes, then the two polynomials have a non-trivial greates common divisor. Implement this algorithm in SymbolicC++ where $a_j, b_j \in \mathbb{Q}$ and apply it to the polynomials

$$p_1(x) = x^3 + 6x^2 + 11x + 6, \qquad p_2(x) = x^2 + 4x + 3.$$

Solution 15. Since $n = 3$ we need the coefficients $c_1, c_2, \ldots, c_5$. Straightforward division yields

$$(x^2 + 4x + 3) : (x^3 + 6x + 11x + 6) = \frac{1}{x} - \frac{2}{x^2} + \frac{4}{x^3} - \frac{8}{x^4} + \frac{16}{x^5} + \cdots$$

Thus

$$c_1 = 1, \quad c_2 = -2, \quad c_3 = 4, \quad c_4 = -8, \quad c_5 = 16.$$

Thus the Hankel matrix is

$$H_3 = \begin{pmatrix} 1 & -2 & 4 \\ -2 & 4 & -8 \\ 4 & -8 & 16 \end{pmatrix}.$$

We find $\det(H_3) = 0$. Thus p_1, p_2 have a greatest common divisior. Obviously

$$(x^2 + 4x + 3)(x + 2) = (x^3 + 6x + 11x + 6).$$

Problem 16. Given an $m \times n$ matrix $A = (a_{jk})$. We define a norm as

$$\|A\| := \max_{1 \le j \le m} \left(\sum_{k=1}^{n} |a_{jk}| \right).$$

Give a C++ implementation using templates.

Solution 16.

Problem 17. Write a C++ program that transposes a square matrix in-place.

Solution 17.

```
// Transpose.cpp
#include <iostream>
using namespace std;

template <class T>
void swap(T* j,T* k)
{
  T temp;
  temp = *j; *j = *k; *k = temp;
}

template <class T>
void transpose(T** a,int rows)
{
  for(int i=0;i<rows;i++)
    for(int j=i+1;j<rows;j++)
       swap(a[i][j],a[j][i]);
}

int main(void)
{
```

```
  int rows = 4; // number of rows = number of columns
  double** m = NULL;
  m = new double* [rows];
  for(int j=0;j<rows;j++) m[j] = new double[rows];
  m[0][0]=2.0;  m[0][1]=2.5;  m[0][2]=3.0; m[0][3]=3.5;
  m[1][0]=1.5;  m[1][1]=1.1;  m[1][2]=1.2; m[1][3]=1.4;
  m[2][0]=1.8;  m[2][1]=5.5;  m[2][2]=0.5; m[2][3]=0.7;
  m[3][0]=0.1;  m[3][1]=7.7;  m[3][2]=0.3; m[3][3]=3.7;
  transpose(m,rows);
  for(int k=0;k<rows;k++)
  {
  for(int l=0;l<rows;l++)
  { cout << "m[" << k << "][" << l << "] = " << m[k][l] << "  "; }
  cout << endl;
  }
  for(int p=0;p<rows;p++) delete[] m[p];
  delete[] m;
  return 0;
}
```

Problem 18. Consider a binary $n \times n$ matrix, where we count the entries from 0. We have $b_{00} = 1$ and $b_{n-1n-1} = 1$. The other 0-1 entries are generated randomly. An ant at entry $(0, 0)$ can only move to the right or down (not diagonal) when this entry contains a 1. Write a C++ program that check whether the ant could reach the entry $(n-1, n-1)$. For example, consider the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

For this case one path

$$(0,0) \to (1,0) \to (1,1) \to (1,2) \to (2,2) \to (3,2) \to (3,3) \to (4,3) \to (4,4)$$

would be possible. Note that the ant could also get stuck at $(2, 0)$.

Solution 18.

Problem 19. Let $z \in \mathbb{C}$ and A be an $n \times n$ matrix over $\mathbb{C}$ with $A^2 = I_n$. Calculate $\exp(zA)$.

Solution 19. Since $A^2 = I_n$ we have

$$\exp(zA) = \cosh(z)I_n + \sinh(z)A.$$

Problem 20. Let $z \in \mathbb{C}$ and A be an $n \times n$ matrix over $\mathbb{C}$ with $A^2 = A$. Calculate $\exp(zA)$.

Solution 20. Since $A^2 = A$ we have $A^n = A$ for all $n \geq 2$. Consequently

$$\exp(zA) = I_n + A(e^z - 1).$$

Problem 21. Apply the Leverrier method to find the determinant of the matrix

$$A(\epsilon) = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & \epsilon \end{pmatrix}$$

where $\epsilon \in \mathbb{R}$. What is the condition on ϵ such that the inverse of $A(\epsilon)$ exists?

Solution 21. We have

$$B_1 = A, \quad p_1 = \mathrm{tr}(B_1) = 2 + \epsilon.$$

Then

$$B_2 = A(B_1 - p_1 I_3) = \begin{pmatrix} -\epsilon & 0 & -1 \\ 0 & -1-\epsilon & 0 \\ -1 & 0 & 1-2\epsilon \end{pmatrix}.$$

Thus $p_2 = \frac{1}{2}\mathrm{tr}(B_2) = -2\epsilon$. For B_3 we obtain

$$B_3 = A(B_2 - p_2 I_3) = \begin{pmatrix} \epsilon - 1 & 0 & 0 \\ 0 & -1+\epsilon & 0 \\ 0 & 0 & \epsilon - 1 \end{pmatrix}$$

and thus $p_3 = \frac{1}{3}\mathrm{tr}(B_3) = \epsilon - 1$ which is the determinant. Consequently the inverse matrix exists if $p_3 \neq 0$, i.e. $\epsilon \neq 1$. Then the inverse is

$$A^{-1} = \frac{1}{\epsilon - 1} \begin{pmatrix} \epsilon & 0 & -1 \\ 0 & \epsilon - 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}.$$

Problem 22. Let

$$A = \begin{pmatrix} 1/4 & 1/2 \\ 1/2 & 1/4 \end{pmatrix}.$$

Let

$$\rho(A) = \max_{1 \leq j \leq 2} |\lambda_j|$$

where λ_j are the eigenvalues of A.

(i) Check that $\rho(A) < 1$.
(ii) If $\rho(A) < 1$, then

$$(I_2 - A)^{-1} = I_2 + A + A^2 + \cdots .$$

Calculate $(I_2 - A)^{-1}$.
(iii) Calculate

$$(I_2 - A)(I_2 + A + A^2 + \cdots + A^k).$$

Solution 22. (i) The eigenvalues are $\lambda_1 = -1/4$ and $\lambda_2 = 3/4$. Thus $\rho(A) < 1$.
(ii) We find

$$(I_2 - A)^{-1} = \begin{pmatrix} 12/5 & 8/5 \\ 8/5 & 12/5 \end{pmatrix}.$$

(iii) We have

$$(I_2 - A)(I_2 + A + A^2 + \cdots + A^k) = I_2 - A^{k+1}.$$

Problem 23. Let A be an $n \times n$ matrix over $\mathbb{R}$. Consider the system of linear equations

$$A\mathbf{x} = \mathbf{b}$$

or

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \qquad i = 1, 2, \ldots, n.$$

We assume that A is invertible. Let $A = C - R$. This is called a splitting of the matrix A and R is the defect matrix of the splitting. Consider the iteration

$$C\mathbf{x}(t+1) = R\mathbf{x}(t) + \mathbf{b}, \quad t = 0, 1, \ldots$$

with a given $\mathbf{x}(0)$.
Let

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -2 & 4 \end{pmatrix}, \quad C = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix}, \quad \mathbf{x}(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Perform the iteration. Does $\mathbf{x}(k)$ $(k = 0, 1, \ldots)$ converge to the solution of $A\mathbf{x} = \mathbf{b}$. Write a C++ program that implements this iteration.

Solution 23. The first iteration yields

$$C\mathbf{x}(1) = R\mathbf{x}(0) + \mathbf{b} \Rightarrow C\mathbf{x}(1) = \mathbf{b} = \begin{pmatrix} 3 \\ 2 \\ 2 \end{pmatrix} \Rightarrow \mathbf{x}(1) = \begin{pmatrix} 3/4 \\ 1/2 \\ 1/2 \end{pmatrix}.$$

The second iteration yields

$$C\mathbf{x}(2) = R\mathbf{x}(1) + \mathbf{b} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} 3/4 \\ 1/2 \\ 1/2 \end{pmatrix} = \begin{pmatrix} 7/2 \\ 13/4 \\ 3 \end{pmatrix}.$$

Thus

$$\mathbf{x}(2) = \begin{pmatrix} 7/8 \\ 13/16 \\ 3/4 \end{pmatrix}.$$

The vector $\mathbf{x}(k)$ converges to the solution $(1, 1, 1)^T$.

Problem 24. Let A be an $n \times n$ matrix over $\mathbb{C}$. Then any eigenvalue of A satisfies the inequality

$$|\lambda| \leq \max_{1 \leq j \leq n} \sum_{k=1}^{n} |a_{jk}|.$$

Write a C++ program that calculates the right-hand side of the inequality for a given matrix. Apply the complex class of STL. Apply it to the matrix

$$A = \begin{pmatrix} i & 0 & 0 & i \\ 0 & 2i & 2i & 0 \\ 0 & 3i & 3i & 0 \\ 4i & 0 & 0 & 4i \end{pmatrix}.$$

Solution 24.

Problem 25. We count the entries of the $n \times n$ matrix from $(0, 0)$ to $(n - 1, n - 1)$. Let $A = (a_{jk})$ be a real $n \times n$ matrix $(j, k = 0, 1, \ldots, n - 1)$. The *permanent* of A is defined to be the real number

$$\operatorname{perm}(A) := \sum_{\sigma \in S_n} \left(\prod_{j \in [n]} A_{j,\sigma(j)} \right)$$

where the summation is over all $n!$ permutations of the set $[n] := \{0, 1, \ldots, n-1\}$. Give an implementation with SymbolicC++ to find the permanent of a given matrix A.

Note that the definition of the determinant for the matrix A is

$$\det(A) := \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \left(\prod_{j \in [n]} A_{j,\sigma(j)} \right)$$

where the summation is over all $n!$ permutations of the set $[n] := \{0, 1, \ldots, n-1\}$ and $\mathrm{sgn}(\sigma)$ equals $+1$ if σ is an even permutation and equals -1 if σ is an odd permutation.

Solution 25.

Problem 26. We count the entries of the $n \times n$ matrix F from $(0,0)$ to $(n-1, n-1)$. We define

$$\omega_n = \exp(i2\pi/n).$$

Now the $n \times n$ matrix F is defined by

$$F = \omega_n^{jk}, \quad j, k = 0, 1, \ldots, n-1.$$

Give a SymbolicC++ implementation for F.

Solution 26.

Problem 27. Consider the linear equation written in matrix form

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

First show that the determinant of the 3×3 matrix is nonzero. Apply two different methods (Gauss elimination and the Leverrier's method) to find the solution. Compare the two methods and discuss.

Solution 27. The determinant of the matrix is $+1$. The Leverrier method provides the inverse matrix

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Applying Gauss elimination provides ...

Problem 28. Consider the two 3×3 permutation matrices (which are of course then also unitary matrices)

$$U_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad U_2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

We want to find efficiently K_1 and K_2 such that $U_1 = e^{K_1}$ and $U_2 = e^{K_2}$. We would apply the spectral decomposition theorem to find K_1, i.e.

$$K_1 = \sum_{j=1}^{3} \ln(\lambda_j) \mathbf{v}_j \mathbf{v}_j^*$$

where λ_j are the eigenvalues of U_1 and $\mathbf{v}_j$ are the corresponding normalized eigenvectors. But then to find K_2 we would apply the property that $U_1^2 = U_2$. Or could we actually apply that $U_2 = U_1^T$? Note that U_1, U_2, I_3 form a commutative subgroup of the group of 3×3 permutation under matrix multiplication.

Solution 28. To apply the spectral theorem we note that the eigenvalues of U_1 are $+1$, $e^{i2\pi/3}$, $e^{i4\pi/3}$. Since $\ln(1) = 0$ we need not to calculate the corresponding eigenvector $\mathbf{v}_1$ since it does not contribute to K_1. For the second and third eigenvalues we obtain the corresponding normalized eigenvectors

$$\mathbf{v}_2 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ e^{i2\pi/3} \\ e^{i4\pi/3} \end{pmatrix}, \quad \mathbf{v}_3 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ e^{i4\pi/3} \\ e^{i2\pi/3} \end{pmatrix}.$$

Now since $\ln(e^{i2\pi/3}) = i2\pi/3$, $\ln(e^{i4\pi/3} = i4\pi/3$ we find for K_1

$$K_1 = i\frac{2\pi}{3}\mathbf{v}_1\mathbf{v}_1^* + i\frac{4\pi}{3}\mathbf{v}_2\mathbf{v}_2^*.$$

Since $U_1^2 = U_2 = e^{K_1}e^{K_1} = e^{2K_1} = e^{K_2}$, the matrix K_2 is given by $K_2 = 2K_1$.

Problem 29. Let A, B, C be $n \times n$ matrices. Simplify

$$(A \otimes I_n \otimes I_n)(I_n \otimes B \otimes I_n(I_n \otimes I_n \otimes C)\,.$$

Solution 29. We have

$$(A \otimes I_n \otimes I_n)(I_n \otimes B \otimes I_n(I_n \otimes I_n \otimes C) = A \otimes B \otimes C.$$

Problem 30. Consider the 3×3 normal matrix

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Study the following four methods to calculate $\exp(A)$. Discuss.
(i) Apply the definition

$$\exp(A) := \sum_{j=0}^{\infty} \frac{A^j}{j!}.$$

(ii) Apply the definition

$$\exp(A) := \lim_{n\to\infty} \left(I_3 + \frac{A}{n}\right)^n.$$

(iii) Apply the spectral theorem, i.e. use the eigenvalues and normalized eigenvectors of A.
(iv) Apply the Cayley-Hamilton theorem which also needs the eigenvalues of A. Keep in mind that one eigenvalue is degenerate.

Solution 30. (i) Since $A^2 = I_3$ and collecting the terms with I_3 and A we obtain

$$e^A = \cosh(1)I_3 + \sinh(1)A.$$

(ii) We have

$$(I_3 + A/2)^2 = I(1 + 1/4) + A.$$
$$(I_3 + A/2)^3 = I(1 + 1/3) + A(1 + 1/27).$$

(iii) The eigenvalues are $\lambda_1 = -1$, $\lambda_2 = 1$, $\lambda_3 = 1$ with the corresponding normalized eigenvectors

$$\mathbf{v}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad \mathbf{v}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{v}_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Thus the spectral decomposition of A is

$$A = \lambda_1 \mathbf{v}_1 \mathbf{v}_1^* + \lambda_2 \mathbf{v}_2 \mathbf{v}_2^* + \lambda_3 \mathbf{v}_3 \mathbf{v}_3^*$$

and $\exp(A)$ follows as

$$\exp(A) = e^{\lambda_1} \mathbf{v}_1 \mathbf{v}_1^* + e^{\lambda_2} \mathbf{v}_2 \mathbf{v}_2^* + e^{\lambda_3} \mathbf{v}_3 \mathbf{v}_3^*.$$

(iv) Using the Cayley-Hamilton theorem we can write since we have a 3×3 matrix

$$f(A) = e^A = a_2 A^2 + a_1 A + a_0 I_3$$

where the complex numbers a_2, a_1, a_0 are determined as follows: Let

$$r(\lambda) := a_2 \lambda^2 + a_1 \lambda + a_0$$

which is the right-hand side of $f(A)$ with A^j replaced by λ^j, where $j = 0, 1, \ldots, n-1$. For each distinct eigenvalue λ_j of the matrix A we consider the equation

$$f(\lambda_j) = r(\lambda_j).$$

Here one eigenvalue has multiplicity two and we have to consider the equation

$$f'(\lambda)|_{\lambda=\lambda_j} = r'(\lambda)|_{\lambda=\lambda_j}.$$

We have the three eigenvalues $\lambda_1 = -1$, $\lambda_2 = +1$, $\lambda_3 = +1$. Thus we obtained the three equations

$$\begin{aligned} e^{-1} &= a_2\lambda_1^2 + a_1\lambda_1 + a_0 = a_2 - a_1 + a_0 \\ e^1 &= a_2\lambda_2^2 + a_1\lambda_2 + a_0 = a_2 + a_1 + a_0 \\ e^1 &= 2a_2\lambda_3 + a_1 = 2a_2 + a_1. \end{aligned}$$

The solution is

$$a_0 = \frac{1}{2}\cosh(1), \quad a_1 = \sinh(1), \quad a_2 = \frac{1}{2}\cosh(1).$$

Consequently

$$\exp(A) = a_2A^2 + a_1A + a_0I_3 = (a_2 + a_0)I_3 + a_1A = \cosh(1)I_3 + \sinh(1)A.$$

Problem 31. Consider an $n \times n$ symmetric tridiagonal matrix over $\mathbb{R}$. Let $f_n(\lambda) := \det(A - \lambda I_n)$ and

$$f_k(\lambda) = \det \begin{pmatrix} \alpha_1 - \lambda & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 - \lambda & \beta_2 & \cdots & 0 \\ 0 & \beta_2 & \ddots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \alpha_{k-1} - \lambda & \beta_{k-1} \\ 0 & \cdots & 0 & \beta_{k-1} & \alpha_k - \lambda \end{pmatrix}$$

for $k = 1, 2, \ldots, n$ and $f_0(\lambda) = 1$, $f_{-1}(\lambda) = 0$. Then

$$f_k(\lambda) = (\alpha - \lambda)f_{k-1}(\lambda) - \beta_{k-1}^2 f_{k-2}(\lambda)$$

for $k = 2, 3, \ldots, n$. Find $f_4(\lambda)$ for the 4×4 matrix

$$\begin{pmatrix} 0 & \sqrt{1} & 0 & 0 \\ \sqrt{1} & 0 & \sqrt{2} & 0 \\ 0 & \sqrt{2} & 0 & \sqrt{3} \\ 0 & 0 & \sqrt{3} & 0 \end{pmatrix}.$$

Solution 31.

=

Chapter 7

Recursion

Problem 1. Let a, b be real positive numbers. If $A = (a + b)/2$ denotes the *arithmetic mean* and $B = \sqrt{ab}$ denotes the *geometric mean*, then $A \geq B$ with equality precisely when $a = b$. Given two positive real numbers, a_0 and b_0, where we suppose that $a_0 \geq b_0$, we consider the recursion

$$a_{j+1} = \frac{1}{2}(a_j + b_j), \qquad b_{j+1} = \sqrt{a_j b_j}$$

so that a_{j+1} is the arithmetic mean of a_j and b_j, while b_{j+1} is their geometric mean. Write a C++ program that implements this recursion with $a_0 = 2.0$ and $b_0 = 1.0$.

Solution 1.

```
// arithgeom.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
  double a0 = 2.0, b0 = 1.0;
  double a1, b1;
  double eps = 0.000000001;
  while((a0-b0) > eps)
  {
  a1 = (a0 + b0)/2.0;
  b1 = sqrt(a0*b0);
```

```
  a0 = a1; b0 = b1;
  }
  cout << "a0 = " << a0 << endl;
  cout << "b0 = " << b0 << endl;
  return 0;
}
```

Problem 2. For all $p > 1$, the iteration $(k = 0, 1, 2, \ldots)$

$$x_{k+1} = \frac{1}{p}((p-1)x_k + x_k^{1-p}a), \qquad x_0 = 1$$

converges quadratically to $a^{1/p}$ if a belongs to

$$a \in \{\, z \in \mathbb{C} : \Re(z) > 0 \text{ and } |z| \leq 1 \,\} \cup \mathbb{R}^+.$$

Write a C++ program that implements this iteration for $p = 3$ and $a = 27$.

Solution 2.

```
// root.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
  double a = 27.0;
  double x0 = 1.0;
  double p = 3.0;
  double x1 = ((p-1.0)*x0 + a*pow(x0,1.0-p))/p;
  double eps = 0.000001;
  while(fabs(x0-x1) > eps)
  {
  x0 = x1;
  x1 = ((p-1.0)*x0 + a*pow(x0,1.0-p))/p;
  }
  cout << "x1 = " << x1;
  return 0;
}
```

Problem 3. Let $0 < x < 2$. The computation of $1/x$ can be done with addition and multiplication using the following recurrence relation

$$a_{j+1} = a_j(1 + c_j), \qquad c_{j+1} = c_j^2 \tag{1}$$

with $j = 0, 1, 2, \ldots$ and the initial values $a_0 = 1$, $c_0 = 1 - x$.
(i) Show that

$$a_j = \frac{1 - c_j}{x} \tag{2}$$

and since $c_j = c_0^{2^j}$ with $|c_0| < 1$, it follows that

$$\lim_{j \to \infty} a_j = \frac{1}{x}.$$

(ii) Write a C++ program that implements this recurrence relation.

Solution 3. (i) To derive (2) from (1) we use the relations

$$a_{j+1} = (1 + c_j)(1 + c_{j-1}) \cdots (1 + c_1)(1 + c_0)$$
$$\frac{1 + c_j}{1 - c_{j+1}} = \frac{1}{1 - c_j}.$$

(ii) Using a `do-while` loop the C++ program is given by

```
// divide1x.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
  double x = 0.5;
  double a0, a1, c0, c1;
  a0 = 1.0; c0 = 1.0 - x;
  double eps = 0.00001;
  double t0, t1;
  do
  {
  a1 = a0*(1.0 + c0); c1 = c0*c0;
  t0 = a0; t1 = a1;
  a0 = a1; c0 = c1;
  } while(fabs(t1-t0) > eps);
  cout << "1/x = " << a1;
  return 0;
}
```

Problem 4. Find a recursion for

$$I_n = \int_0^{\pi/4} \tan^n(x) dx$$

where $n = 0, 1, \ldots$ and

$$I_0 = \int_0^{\pi/4} dx = \frac{\pi}{4}$$

$$I_1 = \int_0^{\pi/4} \tan(x)dx = -\ln(\cos(x))|_0^{\pi/4} = -\ln(\cos(\pi/4)) + \ln(\cos(0)) = \ln(\sqrt{2}).$$

Solution 4. We have

$$\begin{aligned} I_n + I_{n+2} &= \int_0^{\pi/4} (\tan^n(x) + \tan^{n+2}(x))dx = \int_0^{\pi/4} \tan^n(x)\sec^2(x)dx \\ &= \int_0^1 u^n du \\ &= \frac{1}{n+1} \end{aligned}$$

where we used the identity $1+\tan^2(x) \equiv \sec^2(x)$ and the substitution $u = \tan(x)$ with $\tan(0) = 0$ and $\tan(\pi/4) = 1$.

Problem 5. Let $x \in [0, 1]$. Then $\sqrt{x}$ can be approximated by the sequence of polynomials

$$p_{k+1}(x) = p_k(x) + \frac{1}{2}(x - (p_k(x))^2), \qquad k = 0, 1, 2, \ldots$$

and $k \to \infty$ the sequences converges pointwise to $\sqrt{x}$ with $p_0(x) = x$. Write a C++ program that implements this sequence to find an approximation for $\sqrt{x}$.

Solution 5. Using a `while` loop we have the implementation

```
// sqrtx.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
 double x = 0.7;
 double p0 = x;
 double p1 = x+0.5*(x-x*x);
 double eps = 0.00001;
 while(fabs(p1-p0) > eps) { p0 = p1; p1 = p0+0.5*(x-p0*p0); }
 cout << "p1 = " << p1 << endl;
 return 0;
}
```

Problem 6. The number $\pi/2$ can be calculated using the iteration

$$x_{k+1} = x_k y_k, \qquad y_{k+1} = \sqrt{2y_k/(y_k+1)}, \quad k = 0, 1, 2, \ldots$$

where $x_0 = 1$, $y_0 = \sqrt{2}$. Then

$$\lim_{k\to\infty} x_k = \frac{\pi}{2}.$$

Write a C++ program that implements this iteration and thus finds an approximation of $\pi/2$.

Solution 6. Using a `do-while` we have

```
// pihalf.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
 double x0 = 1.0;
 double y0 = sqrt(2.0);
 double x1, y1, t;
 double eps = 0.0001;

 do
 {
 x1 = x0*y0;
 y1 = sqrt(2.0*y0/(y0+1.0));
 t = fabs(x0-x1);
 x0 = x1; y0 = y1;
 } while(t > eps);
 cout << "x1 = " << x1;
 return 0;
}
```

Problem 7. The number π can be calculated using the iteration

$$x_{k+1} = \frac{2x_k y_k}{x_k + y_k}, \qquad y_{k+1} = \sqrt{x_{k+1} y_k}, \quad k = 0, 1, 2, \ldots$$

where

$$x_0 = 2\sqrt{3}, \qquad y_0 = 3.$$

Then

$$\lim_{k\to\infty} x_k = \lim_{k\to\infty} y_k = \pi.$$

Write a C++ program that implements this iteration and thus finds an approximation of π.

Solution 7. Using a `do-while` loop we have

```
// pipfaff.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
 double x0 = 2.0*sqrt(3.0);
 double y0 = 3.0;
 double x1, y1, t;
 double eps = 0.00001;

 do
 {
 x1 = 2.0*x0*y0/(x0 + y0); y1 = sqrt(x1*y0);
 t = fabs(x0-x1);
 x0 = x1; y0 = y1;
 } while(t > eps);
 cout << "x1 = " << x1 << endl;
 cout << "y1 = " << y1 << endl;
 return 0;
}
```

Problem 8. (i) Let r be a real nonzero number. Then $1/r$ can be calculated to ever increasing accuracy applying the map

$$x_{t+1} = x_t(2 - rx_t), \qquad t = 0, 1, \ldots$$

provided the initial estimate x_0 is sufficientlty close to $1/r$. The number of digits of accuracy approximately doubles with each iteration. First find the fixed points of the map. Let $r = 3$ and $x_0 = 2$. Find x_1, x_2, ... Discuss.
(ii) The same iteration can be applied to find the multiplicative inverse modulo any power of 2. For example, to find the multiplicative inverse of 5, modulo 256, we start with $x_0 = 1$ (any odd number will do). Then

$$\begin{aligned} x_1 &= x_0(2 - 5 \cdot x_0) = -3 \\ x_2 &= -3(2 - 5 \cdot (-3)) = -51 \\ x_3 &= -51(2 - 5 \cdot (-51)) = -13107. \end{aligned}$$

Thus $-13107 \bmod 256 = 205$. Thus the multiplicative inverse of 5 (modulo 256) is 205. Write a C++ program that implements this algorithm.

Solution 8.

Problem 9. According to Gauss, the elliptic integral

$$I = \frac{2}{\pi} \int_0^{\pi/2} \frac{dx}{(a^2 \cos^2(x) + b^2 \sin^2(x))^{1/2}}$$

is equal to the limit of any of the two convergent sequences

$$s_0, s_1, s_2, \ldots \quad \text{or} \quad t_0, t_1, t_2 \ldots$$

as defined by the recurrence relations for $j > 0$

$$\begin{aligned} s_{j+1} &= (s_j + t_j)/2 \\ t_{j+1} &= \sqrt{s_j t_j} \end{aligned}$$

and $s_0 = a$, $t_0 = b$. The calculation of the two sequences is called the arithmetic-geometric mean method. Write a C++ program that implements this method.

Solution 9.

Problem 10. Given the sequence of terms

$$t_0,\ t_1,\ t_2,\ \ldots$$

the series of partial sums

$$s_0,\ s_1,\ s_2,\ \ldots$$

is defined such that

$$s_j := t_0 + t_1 + \cdots + t_j \qquad j = 0, 1, 2, \ldots$$

If the sequence is given by the recurrence relation

$$t_{j+1} = f(t_j) \quad \text{for} \quad j \geq 0$$

then the series s_j is determined by

$$\begin{aligned} s_{j+1} &= s_j + t_{j+1} \quad \text{for } j \geq 0 \\ s_0 &= t_0. \end{aligned}$$

Give a C++ implementation for the sequence s_j and the recursion

$$t_{j+1} = f(t_j) = \frac{t_j}{j+1}.$$

Solution 10.

Problem 11. Let m be a positive integer and x be a fixed real number. Then we can calculate $\cos(mx)$ using the recursion

$$\cos((m+1)x) = 2\cos(x)\cos(mx) - \cos((m-1)x)$$

Write a C++ program that implements this recursion. Use the values $m = 10$ and $x = 0.1$.

Solution 11.

Problem 12. Let k be a positive integer $k \geq 2$. Consider the recursion

$$x_{t+1} = x_t + ky_t, \qquad y_{t+1} = x_t + y_t$$

where $t = 0, 1, 2, \ldots$ and $x_0 = y_0 = 1$. Study x_{t+1}/y_{t+1} for $t \to \infty$ and $k = 5$.

Solution 12.

Problem 13. Give a C++ implementation of the recursion

$$c_0 = 1, \quad c_1 = -1, \quad c_{n+1} = -\sum_{k=1}^{n}\sum_{i=0}^{k} c_i c_{k-i} c_{n-k+1}.$$

Solution 13. For c_2 we have $c_2 = -2$.

=

Problem 14. Give a C++ implementation of the recursion

$$x_j(t+1) = (1-\epsilon)x_j(t) + \frac{1}{2}(x_{j-1}(t) + x_{j+1}(t)), \quad t = 0, 1, 2, \ldots$$

where $j = 0, 1, 2, 3$ and $-1 \equiv 3$, $4 \equiv 0$.

Solution 14.

Problem 15. Consider the function

$$f_n(x) = \int_0^\infty y^n \exp(-y^4 - xy^2)dy, \quad n = 0, 1, \ldots$$

(i) Show that

$$f_{n+4}(x) = \frac{n+1}{4} f_n(x) - \frac{x}{2} f_{n+2}(x)$$

using integration by parts.

(ii) Show that

$$\frac{df_n(x)}{dx} = -f_{n+2}(x).$$

Solution 15.

Problem 16. (i) Consider the recursion

$$x_{t+1} = x_t + 5y_t$$
$$y_{t+1} = x_t + y_t$$

where $t = 0, 1, \ldots$. Let $x_0 = y_0 = 1$. Calculate x_1, y_1, x_2, y_2, x_3, y_3 and x_0/y_0, x_1/y_1, x_2/y_2, x_3/y_3.
(ii) Define

$$z_t := \frac{x_t}{y_t}.$$

Find the recursion for z_t. Find the fixed points of this recursion. Are the fixed point stable? Find

$$\lim_{t\to\infty} z_t$$

with $z_0 = x_0/y_0 = 1$.

Solution 16. (i) We have $x_1 = 6$, $y_1 = 2$, $x_2 = 16$, $y_2 = 8$, $x_3 =$, $y_3 =$.
(ii) From

$$\frac{x_{t+1}}{y_{t+1}} = \frac{x_t + 5y_t}{x_t + y_t}$$

we find

$$z_{t+1} = \frac{z_t + 5}{z_t + 1}.$$

Thus the fixed points follows from the solution of the equation

$$\frac{z+5}{z+1} = z$$

with $z = \sqrt{5}$ since x_0 and y_0 are positive.

Problem 17. Let n be a positive integer. A *Dyk word* is a string of length $2n$ with n x's and n y's such that no initial segment of the string of length $2n$ has more y's than x's.
(i) Give the Dyk words for $n = 1$, $n = 2$ and $n = 3$.
(ii) Describe an algorithm to generate the Dyk words for a given n. Give a recursion.

Solution 17. (i) For $n = 1$ we have xy. For $n = 2$ we have $xxyy$, $xyxy$ and for $n = 3$ we have six strings

$$xxxyyy, \quad xyxxyy, \quad xyxyxy, \quad xxyyxy, \quad xxyxyy\,.$$

The number of strings is given by the Catalan numbers.
(ii) A recursion for the Dyk words w is given by

$$w = xw_1yw_2$$

with (possible empty) Dyk words w_1 and w_2.

Problem 18. Let k and n be positive integers. Implement in C++ the recursive function

$$p(k,n) = \begin{cases} 0 & \text{if} \quad k > n \\ 1 & \text{if} \quad k = n \\ p(k+1,n) + p(k,n-k) & \text{otherwise} \end{cases}$$

where $p(1,1) = 1$.

Solution 18.

Problem 19. Consider the alphabet $\{A, B, C\}$ and the Fredholm subsitution

$$A \mapsto AB, \quad B \mapsto BC, \quad C \mapsto CC.$$

Start of with A and find the sequence. Then set $A = C = 0$ and $B = 1$ and find the bitstring.

Solution 19. We have

$$A \mapsto AB \mapsto ABBC \mapsto ABBCBCCC \mapsto \cdots$$

with the bitstrings

$$0 \mapsto 01 \mapsto 0110 \mapsto 01101000 \mapsto \cdots$$

Problem 20. Let $k = 0, 1, \ldots$ and $\phi \in \mathbb{R}$. Consider the integral defined by

$$I_k(\phi) := \int_0^\pi \frac{\cos(k\theta) - \cos(k\phi)}{\cos(\theta) - \cos(\phi)} d\theta\,.$$

Show that $I_0(\phi) = 0$ and $I_1(\phi) = \pi$. Show that $I_k(\phi)$ satisfies the second order difference equation

$$I_{k+2}(\phi) - 2\cos(\phi) I_{k+1}(\phi) + I_k(\phi) = 0, \quad k = 0, 1, \ldots$$

with $I_0(\phi) = 0$ and $I_1(\phi) = \pi$. Solve the second order difference equation.

Solution 20.

=

Problem 21. Let $m = 1, 2, \ldots$. Consider the recursion

$$c_m = \sum_{k=1}^{m} c_{k-1} c_{m-k}$$

with $c_0 = 1$. Define a *generating function*

$$P(x) = \sum_{m=0}^{\infty} c_m x^m = c_0 + c_1 x^1 + c_2 x^2 + \cdots \equiv 1 + c_1 x + c_2 x^2 + \cdots.$$

Then

$$P(x) = 1 + \sum_{m=1}^{\infty} \sum_{k=1}^{m} c_{k-1} c_{m-k} x^m = 1 + x \sum_{k=1}^{\infty} \sum_{m=k}^{\infty} c_{m-k} x^{m-k} c_{k-1} x^{k-1} = 1 + x P^2(x)$$

with the solution for $P(x)$

$$P(x) = \frac{1 - \sqrt{1 - 4x}}{2x}$$

with $P(0) = 1$. Taylor expansion of $P(x)$ provides

$$c_m = \frac{4n - 2}{n + 1} c_{m-1}, \quad m = 1, 2, \ldots$$

and thus ($c_0 = 1$)

$$c_m = \frac{(2m)!}{m!(m+1)!}.$$

Give an implementation of the recursion for c_m and of the two last equations for c_m using SymbolicC++ and the `Verylong` of SymbolicC++.

Solution 21.

Problem 22. Consider the tridiagonal $n \times n$ matrix

$$A = \begin{pmatrix} a_1 & b_2 & 0 & \ldots & 0 & 0 \\ c_2 & a_2 & b_3 & \ldots & 0 & 0 \\ 0 & c_3 & a_3 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & a_{n-1} & b_n \\ 0 & 0 & 0 & \ldots & c_n & a_n \end{pmatrix}$$

with $a_1, a_j, b_j, c_j \in \mathbb{C}$ $(j = 1, 2, \ldots, n)$. It has in general n complex eigenvalues the n roots of the characteristic polynomial $p(\lambda)$. Show that this polynomial can be evaluated by the recursive formula

$$\begin{aligned} p_k(\lambda) &= (\lambda - a_k)p_{k-1} - b_k c_k p_{k-2}(\lambda), \quad k = 2, 3, \ldots, n \\ p_1(\lambda) &= \lambda - a_1 \\ p_0(\lambda) &= 1. \end{aligned}$$

Solution 22.

Problem 23. Let $I_\nu(x)$ be the modified Bessel function. For the asymptotic expansion we have

$$\frac{I_{\nu+1}(x)}{I_\nu(x)} \sim \sum_{j=0}^{\infty} c_j x^{-j}$$

The expansion coefficients c_j obey the quadratic recursive relation

$$c_0 = 1, \quad c_1 = -\left(\nu + \frac{1}{2}\right), \quad c_2 = c_3 = \frac{1}{2}\left(\nu^2 - \frac{1}{4}\right),$$

$$2c_j = (j-1)c_{j-1} - \sum_{\ell=2}^{j-2} c_\ell c_{j-\ell}, \quad j \geq 4.$$

Give a SymbolicC++ implementation of this recursion applying the `Verylong` class. Then apply it to $\nu = 1/2$ and $\nu = -1/2$.

Solution 23.

Problem 24. Let $a > 0$. Give a SymbolicC++ implementation of the recursion

$$\begin{aligned} c_0 &= 1 \\ c_1 &= -(a + 1/2) \\ c_2 &= c_3 = \frac{1}{2}(a^2 - 1/4) \\ c_n &= \frac{1}{2}(n-1)c_{n-1} - \frac{1}{2}\sum_{\ell=2}^{n-2} c_\ell c_{n-\ell}, \quad n \geq 4. \end{aligned}$$

Solution 24. in book 2

Problem 25. Let $j = 2, 3, \ldots$ and

$$P_{j+1} = \frac{P_j^2}{P_{j-1}} + P_j$$

with $P_1 = 1$ and $P_2 = 2$. Give a SymbolicC++ implementation utilizing the class `Verylong`.

Solution 25.

Problem 26. Let $x_1 = \sqrt{1} = 1$, $x_2 = \sqrt{1+\sqrt{2}}$. Study the recurrence relation

$$x_{t+2} = \sqrt{1+\sqrt{2+x_t}}, \quad t = 1, 2, \ldots$$

Find $\lim_{t\to\infty} x_t$.

Solution 26.

Chapter 8

Numerical Techniques

Problem 1. Calculating $\sqrt{2}$ an elementary and ancient recursion consists of the double sequence

$$p_{j+1} = p_j + 2q_j, \qquad q_{j+1} = p_j + q_j$$

with $j = 0, 1, \ldots$ and

$$\lim_{j\to\infty} \frac{p_j}{q_j} = \sqrt{2}.$$

(i) Calculate the first three terms with the initial values $p_0 = q_0 = 1$.
(ii) Give an error estimation with $\epsilon_j := |\sqrt{2} - p_j/q_j|$.
(iii) Write the problem in matrix notation and solve it.

Solution 1. (i) We obtain

$$x_0 = \frac{p_0}{q_0} = 1, \quad x_1 = \frac{p_1}{q_1} = \frac{3}{2}, \quad x_2 = \frac{p_2}{q_2} = \frac{7}{5}, \quad x_3 = \frac{p_3}{q_3} = \frac{17}{12}.$$

The approximation $x_3 = 17/12$ was used by Mesopotamians to replace $\sqrt{2}$.
(ii) We find $\epsilon_{j+1} < \epsilon_j/5$ which gives a geometrical convergence.
(iii) In matrix form we have

$$\begin{pmatrix} p_{j+1} \\ q_{j+1} \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} p_j \\ q_j \end{pmatrix}$$

so that

$$\begin{pmatrix} p_j \\ q_j \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}^j \begin{pmatrix} p_0 \\ q_0 \end{pmatrix}$$

where $j = 1, 2, \ldots$.

Problem 2. Let $a > 0$. Find an iteration to approximate $\sqrt{a}$. Use the fact that if x $(x > 0)$ is the actual square root of a, then $x = a/x$; that is, the two factors x and a/x are equal.

Solution 2. If x is an underestimate of the square root, then a/x is an overestimate, and vice versa. The arithmetic mean of the underestimate and the overestimate is certainly a better estimate than at least of one of them, and hopefully it is better than both. Thus we define the iteration

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \qquad n = 0, 1, 2, \ldots .$$

The sequence of iterates converges quadratically to $\sqrt{a}$.

Problem 3. For any positive number h we define an operator S_h which replaces a continuous function $f : \mathbb{R} \to \mathbb{R}$ by its average over an interval with length h and centre x

$$(S_h f)(x) := \frac{1}{h}\int_{x-h/2}^{x+h/2} f(t)dt.$$

We find that

$$\lim_{h\to 0}(S_h f)(x) = f(x).$$

(i) Show that the operator S_h is linear.
(ii) Show that the operator S_h leaves linear functions unchanged.
(iii) Calculate $S_h f$ for $f(t) = e^{-|t|}\sin(t)$ with $h = 0.1$.

Solution 3. (i) We have

$$\begin{aligned} S_h(c_1 f_1 + c_2 f_2) &= \frac{1}{h}\int_{x-h/2}^{x+h/2} (c_1 f_1(t) + c_2 f_2(t))dt \\ &= c_1 \frac{1}{h}\int_{x-h/2}^{x+h/2} f_1(t)dt + c_2 \frac{1}{h}\int_{x-h/2}^{x+h/2} f_2(t)dt \\ &= c_1(S_h f_1) + c_2(S_h f_2). \end{aligned}$$

(ii) Straighforward calculation yields

$$S_h(c_1 x + c_2) = c_1 x + c_2.$$

(iii) We have

$$(S_{0.1} f)(x) = \frac{1}{h}\int_{x-0.05}^{x+0.05} e^{-|t|}\sin(t)dt =$$

Problem 4. (i) Provide a fast algorithm to calculate $\sqrt{2}$.
(ii) Provide a fast algorithm to calculate the golden mean number $\varphi = (1+\sqrt{5})/2$.

Solution 4. (i) Since $\sqrt{2} \approx 1.4$ we use

$$\sqrt{2} = \frac{7}{5}\frac{5\sqrt{2}}{7} = \frac{7}{5}\sqrt{\frac{50}{49}} = \frac{7}{5}\left(\frac{49}{50}\right)^{-1/2} = \frac{7}{5}\left(\frac{50-1}{50}\right)^{-1/2} = \frac{7}{5}\left(1-\frac{1}{50}\right)^{-1/2}$$

and the expansion

$$(1-x)^{-m} = 1 + mx + \frac{m(m+1)}{2}x^2 + \cdots$$

with $x = 1/50$ and $m = 1/2$.
(ii) We use

$$\varphi = \frac{1}{2} + \frac{\sqrt{5}}{2} = \frac{1}{2} + \left(\frac{4}{5}\right)^{-1/2} = \frac{1}{2} + \left(1-\frac{1}{5}\right)^{-1/2}$$

and the expansion

$$(1-x)^{-m} = 1 + mx + \frac{m(m+1)}{2}x^2 + \cdots$$

where $x = 1/5$ and $m = 1/2$.

Problem 5. If a beam runs into an obstacle a part of the signal is transmitted the rest reflected. The difference between the frequency of the reflected part η and the initial frequency η_0 is the so-called *Doppler shift* $\Delta\eta$ caused by a particle moving with velocity ν in the direction opposite to the transmitted signal. It can be calculated as

$$\Delta\eta = \eta - \eta_0 = \frac{2c\eta_0\nu}{c^2-\nu^2}$$

where c denotes the velocity of sound within the medium. We have $\nu \ll c$. Can the calculation be simplified?

Solution 5. We have

$$\Delta\eta = \frac{2\eta_0\nu}{c}$$

where we used that

$$\frac{2c\eta_0\nu}{c^2-\nu^2} \equiv \frac{2c\eta_0\nu}{c^2(1-\nu^2/c^2)} \equiv \frac{2\eta_0\nu}{c(1-\nu^2/c^2)}$$

and $1-\nu^2/c^2 \approx 1$.

Problem 6. Given pairs of single precision numbers (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4). Decide whether the line segement $(x_1, y_1) - (x_2, y_2)$ intersects the line

determined by $(x_3, y_3)-(x_4, y_4)$ at a unique point. If so, compute the coordinates (x, y) of the intersection point accurate to single precision. That is, if the exact intersection point is (x^*, y^*), then return a point (x, y) such that x is the nearest single precision number to x' and y is the nearest single precision number to y'.

Solution 6. Writing the intersection problem as a system of two linear equations, we find that the line segment intersects the line at a unique point if and only if $d \neq 0$, where

$$d = (y_4 - y_3)(x_1 - x_2) + (x_3 - x_4)(y_1 - y_2).$$

In this case, the coordinates of the intersection point are given by

$$x = \frac{1}{d}((x_1 - x_2)t + (x_3 - x_4)s), \qquad y = \frac{1}{d}((y_1 - y_2)t + (y_3 - y_4)s)$$

where $s := (x_2y_1 - x_1y_2)$ and $t := (x_3y_4 - x_4y_3)$. We can compute d by rewriting the previous expression as a sum of products, multiplying each term using double precision, and then distilling the terms to obtain an expansion for d. Then d vanishes if and only if the first component of the expansion vanishes, and otherwise this component is accurate to double precision. Similarly we can express the numerators of the expressions for x and y as sums of products, then multiply these terms, distill, and use the leading components, which are again accurate to double precision. Taking the quotient of these leading components with the leading component of d gives coordinates which are accurate to near double precision, and at any rate sufficiently accurate to round to the correct single precision quantities, provided no underflow occurs.

Problem 7. The *harmonic series* can be approximated by

$$\sum_{j=1}^{n} \frac{1}{j} \approx 0.5772 + \ln(n) + \frac{1}{2n}.$$

Calculate the left and right hand side for $n = 1$ and $n = 10$.

Solution 7. For $n = 1$ we obtain for the left hand side 1 and for the right-hand side 1.0772 since $\ln(1) = 0$.

Problem 8. Consider the linear first order delay-differential equation

$$\frac{du}{dt} = -u(t - 1).$$

We can find the solution with the ansatz

$$u(t) = Ce^{\lambda t}.$$

Since $du/dt = C\lambda e^{\lambda t}$ and $u(t-1) = Ce^{\lambda(t-1)}$ we obtain

$$\lambda = -e^{-\lambda}.$$

There is no solution if λ is real. If λ is complex then there are an infinite number of solutions. They are given by the *Lambert W function*. Write a C++ program that finds some of the solutions.

Solution 8.

Problem 9. Let $a, b, c, d > 0$ and $a + b + c > d$. Consider the problem of relating the input and output crank angles of a four-bar mechanism. The angles θ and ϕ, respectively, are measured from the line of the fixed pivots. The moving links of fixed length are a, b, c. The fixed link is d. This provides us with the equation

$$b^2 = c^2 + d^2 + a^2 - 2dc\cos(\phi) - 2ac\cos(\phi)\cos(\theta) - 2ac\sin(\phi)\sin(\theta) + 2ad\cos(\theta).$$

With $C_1 = d/c$, $C_2 = d/a$, $C_3 = (d^2 + a^2 - b^2 + c^2)/(2ac)$ we obtain the *Freudenstein equation*

$$C_1\cos(\theta) - C_2\cos(\phi) + C_3 - \cos(\theta - \phi) = 0.$$

Set $a = 1$, $b = c = d = 2$. Solve this transcendental equation with the Newton method for different fixed θ and solve for ϕ.

Solution 9. The derivative of

$$f(\theta, \phi) =$$

with respect to ϕ is given by

$$\frac{df}{d\phi} = R_2\sin(\phi) - \sin(\theta - \phi).$$

Some values for solution

```
theta   phi
0        41.4
5        43.1
10       45
15       46.9
20       48.9
40       57.9
60       67.7

200      56.0
225      39.7
250      32.0
```

Problem 10. Let $f : \mathbb{R} \to \mathbb{R}$ be an analytic function. We define

$$\Delta f(x) := \frac{f(x-\epsilon) + f(x+\epsilon) - 2f(x)}{\epsilon^2}.$$

(i) Let $f(x) = x^2$. Find $\Delta f(x)$. Study $\epsilon \to 0$.
(ii) Give a C++ implementation for $\Delta f(x)$.
(iii) Calculate the derivative of $f(x) = x^2$ using

$$f'(x) := \lim_{\epsilon \to 0} \frac{-11f(x) + 18f(x+\epsilon) - 9f(x+2\epsilon) + 2f(x+3h)}{6\epsilon}.$$

Solution 10. (i) We obtain

$$f(x-\epsilon) + f(x+\epsilon) - 2f(x) = (x-\epsilon)^2 + (x+\epsilon)^2 - 2x^2 = 2\epsilon^2.$$

Thus

$$\Delta f(x) = 2.$$

(ii)
(iii) We find

$$\begin{aligned} f(x+\epsilon) &= (x+\epsilon)^2 = x^2 + 2\epsilon x + \epsilon^2 \\ f(x+2h) &= (x+2\epsilon)^2 = x^2 + 4\epsilon x + 4\epsilon^2 \\ f(x+3h) &= (x+3h)^2 = x^2 + 6\epsilon x + 9\epsilon^2. \end{aligned}$$

Therefore

$$-11f(x) + 18f(x + epsilon) - 9f(x+2\epsilon) + 2f(x+3\epsilon) = 12\epsilon x.$$

It follows that

$$f'(x) = \lim_{\epsilon \to 0} \frac{12\epsilon x}{6\epsilon} = 2x.$$

Problem 11. Given a sequence of ordered parameters (knots): $(x_0, x_1, \ldots, x_m)$, the ith normalized *B-spline basis function* (B-function) $N_{i,k}$ of order k is defined recursively as

$$N_{i,k}(x) = \begin{cases} 1 & \text{if} \quad x_i \le x < x_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad \text{if } k = 1$$

$$N_{i,k}(x) = \frac{x - x_i}{x_{i+k-1} - x_i} N_{i,k-1}(x) + \frac{x_{i+k} - x}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(x) \qquad \text{if } k > 1$$

with $i = 0, 1, \ldots, k$ and $k < m$. The properties of the B-spline basis functions are:

1) *Partition of unity*, i.e.

$$\sum_{i=0}^{m-k} N_{i,k}(x) = 1.$$

2) Positivity

$$N_{i,k}(x) \geq 0.$$

3) Local support

$$N_{i,k}(x) = 0 \text{ for } x \notin [x_i, x_{i+k}].$$

4) C^{k-2} continuity. If the knots $\{ x_i \}$ are pairwise different from each other, then $N_{i,k}(x) \in C^{k-2}$, i.e., the function $N_{i,k}(x)$ is $(k-2)$ times continuously differentiable.

Let $m = 4$ and

$$x_0 = 0,\ x_1 = 0.5,\ x_2 = 1.0,\ x_3 = 1.5,\ x_4 = 2.0.$$

Let $k = 2$. Find $N_{0,2}(x)$, $N_{1,2}(x)$ and $N_{2,2}(x)$. Draw the functions.

Solution 11. From the definition we obtain

$$N_{0,1}(x) = \begin{cases} 1 & \text{if} \quad 0 \leq x < 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{1,1}(x) = \begin{cases} 1 & \text{if} \quad 0.5 \leq x < 1.0 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{2,1}(x) = \begin{cases} 1 & \text{if} \quad 1.0 \leq x < 1.5 \\ 0 & \text{otherwise} \end{cases}$$

$$N_{3,1}(x) = \begin{cases} 1 & \text{if} \quad 1.5 \leq x < 2.0 \\ 0 & \text{otherwise} \end{cases}$$

Thus

$$\begin{aligned} N_{0,2}(x) &= 2xN_{0,1}(x) + (2-2x)N_{1,1}(x) \\ N_{1,2}(x) &= (2x-1)N_{1,1}(x) + (3-2x)N_{2,1}(x) \\ N_{2,2}(x) &= (2x-2)N_{2,1}(x) + (4-2x)N_{3,1}(x). \end{aligned}$$

Problem 12. Let f be a continuos function in the interval $[a, b]$ $(b > a)$. Then

$$\lim_{n\to\infty} \frac{b-a}{n} \sum_{k=1}^{n} f\left(a + \frac{k(b-a)}{n}\right) = \int_a^b f(x)dx.$$

Implement in C++ the left-hand side for a given f, a, b and n with

```
double integrate(double (*f)(double),double a,double b,int n)
```

Apply it to the function

$$f(x) = \sin(x)$$

and the interval $[0, \pi]$ and to the function

$$f(x) = \exp(x * \ln(x))$$

with the interval $[0, 1]$. Choose $n = 10, 100, 1000000$.

Solution 12.

```
// integrate.cpp

#include <iostream>
#include <cmath>
using namespace std;

double integrate(double (*f)(double),double a,double b,int n)
{
 double step = (b-a)/n;
 double x, sum = 0.0;
 for(x=a;x<=b;x+=step) sum += f(x);
 return sum*step;
}

double xtox(double x)
{
 if(x==0.0) return 1.0;
 return exp(x*log(x));
}

int main(void)
{
 const double pi = 4.0*atan(1.0);
 cout.precision(10);
 // integral from 0 to pi of sin x = -cos pi - (-cos 0) = 2
 cout << integrate(sin,0.0,pi,10) << endl;
 cout << integrate(sin,0.0,pi,100) << endl;
 cout << integrate(sin,0.0,pi,1000000) << endl;
 // integral of x^x from 0 to 1
 cout << integrate(xtox,0.0,1.0,10) << endl;
 cout << integrate(xtox,0.0,1.0,100) << endl;
 cout << integrate(xtox,0.0,1.0,1000000) << endl;
 return 0;
}
/*
1.983523538
```

```
1.999835504
2
0.8877326878
0.7834935879
0.7834305107
*/
```

Problem 13. (i) Use the class `Derive` of SymbolicC++ to find the derivative of

$$y = 2x^3 - 5x - 1$$

at the point $x = 2$. Use the data type `double` for x.
(ii) Use the class `Derive` of SymbolicC++ and the class `complex` over `double` to find the derivative of the complex-valued function

$$w = 2z^2 - 5z - 1$$

at the point $z = i$.

Solution 13. The code is

```
// sderive.cpp

#include <iostream>
#include <complex>
#include "Derive.h"
using namespace std;

int main(void)
{
  Derive<double> x;
  x.set(2.0);
  Derive<double> y = 2.0*x*x*x-5.0*x-1.0;
  cout << "The derivative of y at x = " << x << " is "
       << df(y) << endl;
  cout << "value of y at x = " << x << " is "
       << y << endl;

  Derive<complex<double> > z;
  z.set(complex<double>(0.0,1.0));
  complex<double> five(5.0,0.0);
  complex<double> two(2.0,0.0);
  complex<double> one(1.0,0.0);

  Derive<complex<double> > w = two*z*z-five*z-one;
  cout << "The derivative of w at z = " << z << " is "
       << df(w) << endl;
  cout << "value of w at z = " << z << " is "
```

```
       << w << endl;
  return 0;
}
```

Problem 14. Let r be a real number with $r \neq 0$. Then $1/r$ can be calculated to ever increasing precision by using the iteration

$$x_{t+1} = x_t(2 - rx_t), \qquad t = 0, 1, 2, \ldots$$

provided the initial value x_0 is sufficiently close to $1/r$. The number of digits of precision approximately doubles with each iteration. Write a C++ program to find the inverse of $r = 2$ with the initial value $x_0 = 0.8$. Why does the initial value $x_0 = 1$ not work?

Solution 14. Using a `while` loop we have

```
// Division.cpp

#include <iostream>
#include <cmath>
using namespace std;

int main(void)
{
 double d = 2.0;
 double x0 = 0.8;
 double x1 = x0*(2.0-d*x0);
 double eps = 0.000001;
 while(fabs(x0-x1) > eps)
 { x0 = x1; x1 = x0*(2.0-d*x0); }
 cout << "x1 = " << x1 << endl;
 return 0;
}
```

The initial value $x_0 = 1.0$ is not in the domain of attraction of the map. The map $f(x) = 2x - 2x^2$ has the fixed points $x^* = 0$ and $x^* = 1/2$ given as the solution of the equation $2x^* - 2x^{*2} = x^*$. The fixed point $x^* = 1/2$ is stable. If $x_0 = 1$ we have $x_1 = 0$ (fixed point).

Problem 15. In C and C++ the function `fabs` finds the absolute value of a floating point number. How can `fabs` be replaced by an `if` condition and multiplication by -1.0?

Solution 15. Obviously the `if` condition tests whether the number is negative.

```
// myfabs.cpp
```

```
#include <iostream>
using namespace std;

int main(void)
{
  double x;
  cout << "enter a floating point number: ";
  cin >> x;
  if(x < 0.0) x = -1.0*x;
  cout << "the absolute value is: " << x << endl;
  return 0;
}
```

Problem 16. What is the following code doing

```
// InvSqrt.cpp

#include <iostream>
using namespace std;

float invSqrt(float x)
{
  float xhalf = 0.5f*x;
  int i = *(int*)& x;        // get bits for floating value
  i = 0x5f3759df - (i >> 1); // initial guess
  x = *(float*)& i;          // convert bits back to float
  x *= 1.5f - xhalf*x*x;
  x *= 1.5f - xhalf*x*x;
  return x;
}

int main(void)
{
 float x1 = 10.0f;
 float r1 = invSqrt(x1);
 cout << "r1 = " << r1 << endl;

 float x2 = 100.0f;
 float r2 = invSqrt(x2);
 cout << "r2 = " << r2 << endl;
 return 0;
}
```

Solution 16. The code gives an approximation of the inverse square root $1/\sqrt{x}$ of a floating point number x using the Newton method and a clever initial guess of $y = 1/\sqrt{x}$. We define $F(y) = 1/y^2 - x$. We want the positive root of $F(y) = 0$. Given x choose an initial value y_0. Then the Newton iteration scheme

is

$$y_{n+1} = y_n - \frac{F(y_n)}{F'(y_n)}, \qquad n \geq 0$$

where $F'(y) = -2/y^3$ is the derivative of F. Thus

$$y_{n+1} = \frac{y_n(3 - xy_n^2)}{2}.$$

The initial guess is selected as follows. The IEEE 32-bit float has a mantissa M filling bit positions 0 through 22, an 8-bit biased exponent E filling bits 23 through 30, and a sign bit in position 31. The function `invSqrt` expects nonnegative input, so the sign bit is 0 for the input x. The bias is 127. Thus the true exponent is $e = E - 127$. The corresponding number in readable form is $x = 1.M * 2^e$. We want y_0 to be a good approximation to $1/\sqrt{x} = (1/\sqrt{1.M}) * 2^{-e/2}$. The biased exponent for $-e/2$ is $-e/2 + 127$. In terms of integer arithmetic, this is `0xbe-(E >> 1)` where E is the biased exponent for x. The statement

```
i = 0x5f3759df - (i >> 1);
```

implicitly computes the biased exponent $-e/2 + 127$. This statement also implicitly computes the mantissa for the initial guess y_0.

Problem 17. Consider the function $f : \mathbb{R}^n \to \mathbb{R}$, where it is assumed that the function f is at least twice continuous differentiable. We want to find the minimum of the function f. Let

$$H(\mathbf{x}) := \left(\frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_k} \right)$$

be the *Hessian matrix* and $j, k = 1, 2, \ldots, n$. In the *Levenberg-Marquardt algorithm* we apply

$$\mathbf{x}_{t+1} = \mathbf{x}_t - (H(\mathbf{x}_t) + \lambda \mathrm{diag}(H(\mathbf{x}_t))^{-1} \nabla f(\mathbf{x}_t)$$

where $t = 0, 1, 2, \ldots$, λ is the step length and given initial values. We need matrix inversion as part of the update. Since the determinant of the Hessian matrix is proportional to the curvature of f, the iteration implies a large step in the direction with low curvature (i.e., an almost flat terrain) and a small step in the direction with hight curvature (i.e. a steep incline). Write a C++ program that applies this method to solve the system of equation

$$x_1 = \sin(x_1 + x_2), \qquad x_2 = \cos(x_1 - x_2)$$

by finding the minimum the function

$$f(x_1, x_2) = (x_1 - \sin(x_1 + x_2))^2 + (x_2 - \cos(x_1 - x_2))^2.$$

Use the initial values $x_{1,0} = x_{2,0} = 0$.

Solution 17.

Problem 18. A *polygon* is a closed plane figure with n sides. If all sides and angles are equivalent the polygon is called regular. The area of a planar convex polygon with vertices

$$(x_0, y_0),\ \ (x_1, y_1),\ \ \ldots,\ \ (x_{n-1}, y_{n-1})$$

is given by

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1}y_i), \qquad x_n \equiv x_0,\ \ y_n \equiv y_0.$$

A *polygon* in the plane $\mathbb{R}^2$ is a closed figure with n sides. If all sides and angles are equal the polygon is called regular. The area of a planar convex polygon with vertices

$$(x_0, y_0),\ \ (x_1, y_1),\ \ldots,\ (x_{n-1}, y_{n-1})$$

is given by

$$A = \frac{1}{2}\left|\sum_{j=1}^{n-1}(x_j y_{j+1} - x_{j+1}y_j)\right|, \quad x_n \equiv x_0,\ \ y_n \equiv y_0.$$

(i) Write a C++ program that finds the area of a given planar convex polygon. Apply the modulus operator `%` to identify n and 0.
(ii) Write a Java program that finds the area of a given planar convex polygon. Apply the *modulus operator* `%` to identify n and 0.

Solution 18. In the example the polygon is the unit square. The C++ program is

```
// polygon.cpp

#include <iostream>
using namespace std;

double area(double* x,double* y,int n)
{
  double area = 0.0;
  for(int i=0;i<n;i++)
  { area += (x[i]*y[(i+1)%n]-x[(i+1)%n]*y[i]); }
  return 0.5*area;
}

int main(void)
{
```

```
   int n = 4;
   double* x = new double[n];
   double* y = new double[n];
   x[0] = 0.0; x[1] = 1.0; x[2] = 1.0; x[3] = 0.0;
   y[0] = 0.0; y[1] = 0.0; y[2] = 1.0; y[3] = 1.0;
   double result = area(x,y,n);
   cout << "result = " << result << endl;
   delete[] x; delete[] y;
   return 0;
}
```

The Java program is

```
// Polygon.java

public class Polygon
{
 public static void main(String[] args)
 {
 double[] x = new double[4];
 double[] y = new double[4];
 x[0] = 0.0; x[1] = 1.0; x[2] = 1.0; x[3] = 0.0;
 y[0] = 0.0; y[1] = 0.0; y[2] = 1.0; y[3] = 1.0;
 double area = 0.0;
 int length = x.length;
 for(int j=0;j<x.length;j++)
 { area += (x[j]*y[(j+1)%length]-x[(j+1)%length]*y[j]); }
 area = 0.5*Math.abs(area);
 System.out.println("area of polygon = " + area);
 }
}
```

Problem 19. Given a time series $\{\, x_i \,:\, i = 0, 1, \ldots, N-1 \,\}$, the linear correlation of an epoch consisting of K points $\{\, x_i \,:\, i = 0, 1, \ldots, K-1 \,\}$ and another epoch of the same length K covering a different time interval $\{\, x_k \,:\, k = j, j+1, \ldots, K+j-1 \,\}$, $K + j \leq N$, is given by the *cross-correlation coefficient*

$$r_j := \sum_{i=0}^{K-1} \frac{(x_i - \langle x_0 \rangle)(x_{i+j} - \langle x_j \rangle)}{\sigma_0 \sigma_j}$$

where $\langle x_0 \rangle$ and $\langle x_j \rangle$ are, respectively, the mean values of the epochs starting at x_0 and x_j and σ_0 and σ_j the corresponding standard deviations. Write a C++ program that find the correlation coefficient for the logistic map

$$x_{t+1} = 4x_t(1 - x_t), \qquad t = 0, 1, 2, \ldots$$

and $x_0 = 1/3$. Let $N = 2048$ and $K = 1024$.

Solution 19.

Problem 20. The standard Hermite polynomial satisfy the recursion relations

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \qquad \frac{dH_n(x)}{dx} = 2nH_{n-1}(x)$$

with $H_0(x) = 1$. Combining these two relations we get the recursion relation

$$H_{n+1}(x) = \left(2x - \frac{d}{dx}\right) H_n(x).$$

Write a C++ program using SymbolicC++ which implement this recursion relation.

Solution 20.

Problem 21. Write a C++ program to implement an approximation to the integral

$$\int_0^x \exp(-s^2)ds = x - \frac{x^3}{3 \cdot 1!} + \frac{x^5}{5 \cdot 2!} - \frac{x^7}{7 \cdot 3!} + \cdots$$

Solution 21.

Problem 22. Consider the 3×3 matrix

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Find A^2 and A^3. We know that

$$\mathrm{tr}(A) = \lambda_1 + \lambda_2 + \lambda_3, \quad \mathrm{tr}(A^2) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \quad \mathrm{tr}(A^3) = \lambda_1^3 + \lambda_2^3 + \lambda_3^3.$$

Use Newton's method to solve this system of equations to find the eigenvalues of A.

Solution 22. For the powers of A we have

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}, \quad A^3 = \begin{pmatrix} 2 & 3 & 3 \\ 3 & 2 & 3 \\ 3 & 3 & 2 \end{pmatrix}.$$

Thus we have the equations

$$\begin{aligned} \mathrm{tr}(A) &= 0 = \lambda_1 + \lambda_2 + \lambda_3 \\ \mathrm{tr}(A^2) &= 6 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \\ \mathrm{tr}(A^3) &= 6 = \lambda_1^3 + \lambda_2^3 + \lambda_3^3. \end{aligned}$$

We use the three dimensional Newton-Raphson method to find the solution

$$\lambda_1 = 2, \quad \lambda_2 = -1, \quad \lambda_3 = -1.$$

The following C++ program implements the Newton-Raphson method.

```
// nr3.cpp

#include <iostream>
#include <cmath>
using namespace std;

typedef double dfunction(double[3]);

// f0, f1 and f2 implement the equations
//  x0+x1+x2=0.0
//  x0^2+x1^2+x2^2-6.0=0.0
//  x0^3+x1^3+x2^3-6.0=0.0
double f0(double x[3]) {return x[0]+x[1]+x[2];}
double f1(double x[3]) {return x[0]*x[0]+x[1]*x[1]+x[2]*x[2]-6.0;}
double f2(double x[3])
{return x[0]*x[0]*x[0]+x[1]*x[1]*x[1]+x[2]*x[2]*x[2]-6.0;}

// dfij is the partial derivative of fi with respect to x[j]
double df00(double x[3]) {return 1.0;}
double df01(double x[3]) {return 1.0;}
double df02(double x[3]) {return 1.0;}
double df10(double x[3]) {return 2.0*x[0];}
double df11(double x[3]) {return 2.0*x[1];}
double df12(double x[3]) {return 2.0*x[2];}
double df20(double x[3]) {return 3.0*x[0]*x[0];}
double df21(double x[3]) {return 3.0*x[1]*x[1];}
double df22(double x[3]) {return 3.0*x[2]*x[2];}

// MatrixMul multiplies a 3x3 matrix with a 3 dimensional vector
// and stores the result in ans
void MatrixMul(double A[3][3],double x[3],double ans[3])
{
 ans[0]=A[0][0]*x[0]+A[0][1]*x[1]+A[0][2]*x[2];
 ans[1]=A[1][0]*x[0]+A[1][1]*x[1]+A[1][2]*x[2];
 ans[2]=A[2][0]*x[0]+A[2][1]*x[1]+A[2][2]*x[2];
}

// CalculateInverse inverts a 3x3 matrix using the adjoint matrix
// the return value indicates if the matrix could be inverted
int CalculateInverse(double A[3][3],double Ainv[3][3])
{
 double det=(A[0][0]*A[1][1]*A[2][2]+A[0][1]*A[1][2]*A[2][0]
            +A[0][2]*A[1][0]*A[2][1]-A[0][0]*A[1][2]*A[2][1]
```

```
            -A[0][1]*A[1][0]*A[2][2]-A[0][2]*A[1][1]*A[2][0]);
 if(det==0.0) return 0;
 Ainv[0][0]= (A[1][1]*A[2][2]-A[1][2]*A[2][1])/det;
 Ainv[0][1]=-(A[0][1]*A[2][2]-A[0][2]*A[2][1])/det;
 Ainv[0][2]= (A[0][1]*A[1][2]-A[0][2]*A[1][1])/det;
 Ainv[1][0]=-(A[1][0]*A[2][2]-A[1][2]*A[2][0])/det;
 Ainv[1][1]= (A[0][0]*A[2][2]-A[0][2]*A[2][0])/det;
 Ainv[1][2]=-(A[0][0]*A[1][2]-A[0][2]*A[1][0])/det;
 Ainv[2][0]= (A[1][0]*A[2][1]-A[1][1]*A[2][0])/det;
 Ainv[2][1]=-(A[0][0]*A[2][1]-A[0][1]*A[2][0])/det;
 Ainv[2][2]= (A[0][0]*A[1][1]-A[0][1]*A[1][0])/det;
 return 1;
}

// the Newton-Raphson method
// lambda[3] contains the initial values and stores the final result
// f[3] implements the three equations
// df[3][3] implements the partial derivatives of f[3]
// eps is the required accuracy
// maxiter is the maximum number of iterations
int Newton(double lambda[3],dfunction *f[3],dfunction *df[3][3],
           double eps,int maxiter)
{
 double temp1[3],temp2[3],m[3][3],minv[3][3];
 int i=0;

 while(((fabs(f[0](lambda))>eps) || (fabs(f[1](lambda))>eps)
         || (fabs(f[2](lambda))>eps)) && i<maxiter)
 {
 // calculate the partial derivatives
 m[0][0]=df[0][0](lambda);m[0][1]=df[0][1](lambda);
 m[0][2]=df[0][2](lambda);
 m[1][0]=df[1][0](lambda);m[1][1]=df[1][1](lambda);
 m[1][2]=df[1][2](lambda);
 m[2][0]=df[2][0](lambda);m[2][1]=df[2][1](lambda);
 m[2][2]=df[2][2](lambda);
 // evaluate the functions
 temp1[0]=f[0](lambda); temp1[1]=f[1](lambda);
 temp1[2]=f[2](lambda);
 if(!CalculateInverse(m,minv)) return 0;
 MatrixMul(minv,temp1,temp2);
 lambda[0]-=temp2[0]; lambda[1]-=temp2[1]; lambda[2]-=temp2[2];
 i++;
 }
 if (i>=maxiter) return 0;
 return 1;
}
```

```
int main(void)
{
 dfunction *f[3],*df[3][3];
 // for every one solution found there are two more
 // we choose the initial values to satisfy the first equation
 double lambda[3]={0,1,-1};

 f[0]=f0;f[1]=f1;f[2]=f2;
 df[0][0]=df00;df[0][1]=df01;df[0][2]=df02;
 df[1][0]=df10;df[1][1]=df11;df[1][2]=df12;
 df[2][0]=df20;df[2][1]=df21;df[2][2]=df22;

 if(Newton(lambda,f,df,1e-50,100))
 {
  cout<<"Lambda 1 : "<<lambda[0]<<endl;
  cout<<"Lambda 2 : "<<lambda[1]<<endl;
  cout<<"Lambda 3 : "<<lambda[2]<<endl;
 }
 else cout<<"Could not find a solution "
          <<"with the required accuracy."<<endl;
 return 0;
}
// Output:
// Lambda 1 : -1
// Lambda 2 : 2
// Lambda 3 : -1
```

Problem 23. What is the output of the following C++ code

```
// epsilon.cpp

#include <iostream>
using namespace std;

int main(void)
{
 double eps = 1.0, x = 2.0, y = 1.0;
 while(y < x) { eps *= 0.5; x = 1.0 + eps; }
 eps *= 2.0;
 cout << "eps = " << eps;
}
```

Solution 23. We obtain

```
2.220446049250313e-0.16
```

which is 2^{52} the smallest positive number that satisfies $1.0 + \epsilon > 1.0$.

Problem 24. Use numerical integration to show that

$$\int_0^1 \frac{x}{1+x^2}\ln(1+x)dx = 0.162865007.$$

Solution 24.

Problem 25. Let $x > 0$. Find the solution of the equation

$$(x-2)^2 = \ln(x).$$

First show that the equation has a root in $[1, 2]$.

Solution 25.

Problem 26. Consider the mathematical expression

$$\sin(b) + a * b \underbrace{+} c * d + (a - b).$$

(i) Write this mathematical expression as a binary tree with the root indicated by the underbrace. Then evaluate this binary tree from bottom to top with the values $a = 2$, $b = \pi/2$, $c = 4$, $d = 1$.
(ii) Am alternative to represent a mathematical expression as tree is multiexpression programming. Use multiexpression programming to evaluate the mathematical expression given above.

Solution 26.

Problem 27. Find an approximation of $\sqrt{30}$ utilizing

$$\sqrt{30} = \sqrt{25+5} = 5\sqrt{1+0.2}.$$

Solution 27. We have

$$5\sqrt{1+0.2} \approx 5(1+\frac{1}{2}0.2) = 5.5.$$

Chapter 9

Random Numbers

Problem 1. Calculate the integral

$$\int_0^1 |\cos(2\pi x)|dx$$

using the random numnber generator described in problem 7, chapter 10, page 250, Problems and Solutions in Scientific Computing. Compare to the exact result by solving the integral.

Solution 1. Note that

$$\cos(0) = 1, \quad \cos(\pi/2) = 0, \quad \cos(3\pi/2) = 0, \quad \cos(2\pi) = 1.$$

The integration yields

$$\begin{aligned}\int_0^1 |\cos(2\pi x)|dx &= \int_0^{1/4} \cos(2\pi x)dx - \int_{1/4}^{3/4} \cos(2\pi x)dx + \int_{3/4}^1 \cos(2\pi x)dx \\ &= \frac{1}{2\pi}\sin(2\pi x)\Big|_0^{1/4} - \frac{1}{2\pi}\sin(2\pi x)\Big|_{1/4}^{3/4} + \frac{1}{2\pi}\sin(2\pi x)\Big|_{3/4}^{1} \\ &= \frac{2}{\pi}.\end{aligned}$$

The program is

```
// MonteCarlo.cpp

#include <iostream>
#include <cmath>
```

```
using namespace std;

const double PI = 3.1415926535;

void randgen(double* x)
{ *x = fmod((*x+PI)*(*x+PI)*(*x+PI)*(*x+PI)*(*x+PI),1.0); }

double sum(double (*f)(double),int n)
{
  double u = 0.618;
  double result = 0.0;
  for(int i=0;i<n;i++) { randgen(&u); result += f(u); }
  return result/n;
}

double f(double x) { return fabs(cos(2.0*PI*x)); }

int main(void)
{
  double value;
  int n = 1000;
  value = sum(f,n);
  cout << "value = " << value;
  return 0;
}
```

Chapter 10

Optimization Problems

Problem 1. Consider an overdetermined linear system $A\mathbf{x} = \mathbf{b}$, where A is an $m \times n$ matrix with $m > n$. Thus $\mathbf{x}$ is a column vector with n rows and $\mathbf{b}$ is a column vector with m rows. Write a genetic algorithm program that finds the Chebyshev or *minmax solution* to set of overdetermined linear equations $A\mathbf{x} = \mathbf{b}$, i.e. the column vector $\mathbf{x}$ which minimizes

$$c = \max_{1 \le i \le m} c_i \equiv \max_{1 \le i \le m} \left| b_i - \sum_{j=1}^{n} a_{ij} x_j \right|.$$

Apply the program to the overdetermined linear system

$$\begin{pmatrix} 1 & -1 & 1 \\ 1 & -0.5 & 0.25 \\ 1 & 0 & 0 \\ 1 & 0.5 & 0.25 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \\ 0 \\ 0.5 \\ 2.0 \end{pmatrix}.$$

Solution 1. We have

$$\begin{aligned} c_1 &= |1 - x_1 + x_2 - x_3| \\ c_2 &= |0.5 - x_1 + 0.5x_2 - 0.25x_3| \\ c_3 &= |x_1| \\ c_4 &= |0.5 - x_1 - 0.5x_2 - 0.25x_3| \\ c_5 &= |2 - x_1 - x_2 - x_3|. \end{aligned}$$

The solution is given by

$$x_1 = \frac{1}{6}, \quad x_2 = \frac{1}{3}, \quad x_3 = \frac{4}{3}.$$

Chapter 11

String Manipulations

Problem 1. Given a string of digits with or without a decimal point, for example "345678" or "12.3456". Write a C++ program that converts the string to a floating point number `double`.

Solution 1.

```
// strtodbl.cpp

#include <cassert>
#include <cctype>
#include <iostream>
#include <string>
using namespace std;

double strtodbl(const string &s)
{
 int negative = 0;
 int fraction = 0;
 double value = 0.0;
 double fracpart = 1.0;
 for(int i=0;i<s.length();i++)
 {
 switch(s[i])
 {
 case '-': assert(i==0);    negative = 1; break;
 case '.': assert(!fraction); fraction = 1; break;
 default : assert(isdigit(s[i]));
 if(fraction) value += (s[i]-'0')*(fracpart*=0.1);
```

```
 else         value = value*10 + (s[i]-'0');
 break;
 }
 }
 return value;
}

int main(void)
{
 cout << strtodbl("") << endl;
 cout << strtodbl("12345") << endl;
 cout << strtodbl("12.345") << endl;
 cout << strtodbl("1.2345") << endl;
 cout << strtodbl(".12345") << endl;
 cout << strtodbl("12345.") << endl;
 cout << strtodbl(".") << endl;
 string s = ".123456789012345678901234567890";
 double r = strtodbl(s);
 cout << "r = " << r << endl;
 return 0;
}
/*
0
12345
12.345
1.2345
0.12345
12345
0
0.123457
*/
```

Problem 2. Write a C++ program that implements the *Levenshtein distance* (also called the edit distance).

Solution 2.

```
// levenshtein.cpp

#include <iostream>
#include <string>
using namespace std;

int min(int a,int b,int c)
{
 int m = a;
 if(b < m) { m = b; }
 if(c < m) { m = c; }
```

```
 return m;
}

int LD(string s,string t,int n,int m,int** D)
{
 int cost;  // cost
 // step 1
 if(n==0) return m;
 if(m==0) return n;
 // step 2
 for(int p=0;p<=n;D[p][0]=p++);
 for(int q=0;q<=m;D[0][q]=q++);
 // step 3
 for(int i=1;i<=n;i++) {
 // step 4
 for(int j=1;j<=m;j++) {
 cost = (t.substr(j-1,1)==s.substr(i-1,1) ? 0 : 1);
 // step 6
 D[i][j] = min(D[i-1][j]+1,D[i][j-1]+1,D[i-1][j-1]+cost);
 }
 }
 return D[n][m];
}

int main(void)
{
 string s = "010101010101"; string t = "101010101010";
 int n = s.length(); int m = t.length();
 int** D = NULL;
 D = new int*[n+1];
 for(int k=0;k<=n;k++) D[k] = new int[m+1];
 int distance = LD(s,t,n,m,D);
 cout << "distance = " << distance << endl;
 for(int l=0;l<=n;l++) delete D[l]; delete[] D;
 return 0;
}
```

Problem 3. Write a C++ program that find the first character in an ASCII string that occurs only once. Find a solution that minimizes the number of comparisons between characters. Note that `'\0'` denotes the null character.

Solution 3. We have

```
// first.cpp

#include <iostream>
#include <string>
using namespace std;
```

```
/* Order (s.length())^2 comparisons for the worst case */

char naive(string s)
{
 size_t i, j;
 for(i=0;i<s.length();i++)
 {
  for(j=0; j<s.length(); j++)
   if(i!=j && s[i]==s[j]) break;
  if(j==s.length()) return s[i];
 }
 return '\0';
}

/* Order s.length() comparisons for the worst case */

char optimal(string s)
{
 size_t i, j;
 int ascii[128];
 for(i=0;i<128;i++) ascii[i] = 0;
 for(i=0;i<s.length();i++) ascii[s[i]]++;
 for(i=0;i<s.length();i++) if(ascii[s[i]]==1) return s[i];
 return '\0';
}

int main(void)
{
 char firstn, firsto;
 string s;
 cout << "Enter a string: ";
 getline(cin,s);
 firstn = naive(s);
 firsto = optimal(s);
 if(firstn != firsto)
 {
 cerr << "Error: the naive and optimal algorithms returned different values,"
      << "       '" << firstn << "' and '" << firsto << "' respectively"
      << endl;
  return 1;
 }

 if(firstn=='\0')
  cout << "No character has only one occurence." << endl;
 else cout << "'" << firstn
           << "' is the first character with only one occurence."
           << endl;
```

```
 return 0;
}
```

Problem 4. (i) Write a C++ program that uses the string class and the line

```
string* sa = new string[N];
```

and then concatentes the strings.
(ii) Write a C++ program that uses the string class and the line

```
string* sa = new string(6,' ');
```

and then concatentes the characters.

Solution 4. (i) An example is

```
// stringarray.cpp

#include <iostream>
#include <string>
using namespace std;

int main(void)
{
  int N = 3;
  string* sa = new string[N];
  sa[0] = "a"; sa[1] = "ab"; sa[2] = "aba";
  string s = "";
  for(int i=0;i<N;i++) { s += sa[i]; }
  cout << "s = " << s << endl;
  delete[] sa;
  return 0;
}
```

(ii) An example is

```
// newstring.cpp

#include <iostream>
#include <string>
using namespace std;

int main(void)
{
  string* s = new string(6,' ');
  (*s)[0] = ' '; (*s)[1] = 'I'; (*s)[2] = 'S';
  (*s)[3] = 'S'; (*s)[4] = 'C'; (*s)[5] = ' ';
  cout << *s << endl;
  cout << s << endl;
```

```
  delete s;
  return 0;
}
```

Chapter 12

Programming Problems

Problem 1. The rank of an element in a sequence (one-dimensional array) of numbers is the number of smaller elements in the sequence plus the number of equal elements that appear to its left. For example, if the sequence is given as the one-dimensional array $a = [4, 3, 9, 3, 7]$, then the ranks are $r = [2, 0, 4, 1, 3]$. Write a C++ program with a function `void rank(T* a,int n,int* r)` that computes the ranks of the elements of the array $a[0 : n-1]$. Once the elements have been ranked using the function `rank()` write a function `rearrange()` that rearrange them in nondecreasing order so that $a[0] \leq a[1] \leq \cdots \leq a[n-1]$ by moving elements to positions corresponding to their ranks.

Solution 1. We can estimate the complexity of the function `rank` by counting the number of comparisons between elements of the array `a`. These comparisons are done in the `if` statement. For each value of `i`, the number of element comparisons is `i`. Thus the total number of element comparisons is $1 + 2 + \cdots + n - 1 = (n-1)n/2$.

```
// rank.cpp

#include <iostream>
using namespace std;

template<class T>
void rank(T* a,int n,int* r)
{
  for(int i=0;i<n;i++) r[i] = 0;  // initialize
  // compare all elements
  for(int j=1;j<n;j++)
    for(int k=0;k<j;k++)
```

```
      if(a[k] <= a[j]) r[j]++;
      else r[k]++;
}

template <class T>
void rearrange(T* a,int n,int* r)
{
  T* u = new T[n+1];
  // move to the correct place in u
  for(int i=0;i<n;i++) u[r[i]] = a[i];
   // move back to array a
   for(int j=0;j<n;j++) a[j] = u[j];
  delete[] u;
}

int main(void)
{
  int n = 5;
  int* a = new int[n];
  a[0] = 4; a[1] = 3; a[2] = 9; a[3] = 3; a[4] = 7;
  int* r = new int[n];
  rank(a,n,r);
  for(int i=0;i<n;i++)
  cout << "r[" << i << "] = " << r[i] << endl;
  rearrange(a,n,r);
  for(int j=0;j<n;j++)
  cout << "a[" << j << "] = " << a[j]  << endl;
  delete[] a; delete[] r;
  return 0;
}
```

Problem 2. Given an array of 20 integer numbers. We would like to assign the numbers $0, 4, 8, \ldots, 76$ to the array, i.e.

`array[0] = 0, array[1] = 4, ... , array[19] = 76.`

Can the multiplication `i*4` in the following C++ program `forloop1.cpp` be avoided and replaced by addition? Can the C++ program be even more efficient?

```
// forloop1.cpp

#include <iostream>
using namespace std;

int main(void)
{
  int* array = new int[20];
  for(int i=0;i<20;i++) { array[i] = i*4; }
  delete[] array;
```

```
  return 0;
}
```

Solution 2. In the C++ program `forloop2.cpp` we replace the multiplication by addition.

```
// forloop2.cpp

#include <iostream>
using namespace std;

int main(void)
{
  int* array = new int[20];
  int temp = 0;
  for(int i=0;i<20;i++) { array[i] = temp; temp += 4; }
  delete[] array;
  return 0;
}
```

Using pointers and pointer arithmetic the program can even more efficient.

```
// forloop3.cpp

#include <iostream>
using namespace std;

int main(void)
{
  int* array = new int[20];
  int temp = 0;
  int* p = array;
  for(int i=0;i<20;i++) { *(p++) = temp; temp += 4; }
  delete[] array;
  return 0;
}
```

```
// forloop4.cpp

#include <iostream>
using namespace std;

int main(void)
{
  int* array = new int[20];
  int* p = array;
  for(int temp=0;temp<80;temp+=4) { *(p++) = temp; }
  delete[] array;
  return 0;
}
```

Problem 3. Let n be a positive integer. Then the solutions of $z^n = 1$ are called the roots of unity. For example, if $n = 4$ we have $z_1 = 1$, $z_2 = i$, $z_3 = -1$, $z_4 = -i$. Write a C++ program using the class `complex<double>` that stores the unit roots in an array for a given n.

Solution 3. The roots are given by $(i = \sqrt{-1})$

$$\exp(ij2\pi/n), \qquad j = 0, 1, \ldots, n-1.$$

Now we can write

$$\exp(ij2\pi/n) \equiv \cos(j2\pi/n) + i\sin(j2\pi/n).$$

```
// unitroots.cpp

#include <iostream>
#include <complex>
#include <cmath>
using namespace std;

void unitroots(complex<double>* a,int n)
{
 const double pi = 3.14159265358979323846;
 for(int j=0;j<n;j++)
  a[j] = complex<double>(cos(j*2.0*pi/n),sin(j*2.0*pi/n));
}

int main(void)
{
 int n = 4;
 complex<double>* a = new complex<double>[n];
 unitroots(a,n);
 for(int j=0;j<n;j++)
 {
 cout << "real part a[" << j << "] = " << a[j].real() << endl;
 cout << "imag part a[" << j << "] = " << a[j].imag() << endl;
 }
 delete[] a;
 return 0;
}
```

Problem 4. Extend the C++ program `Gauss0.cpp` to a template basis so that it can also be used for other data types such as `Rational<int>` and `Rational<Verylong>`.

Solution 4. The file `identity.h` is a header file provided in SymbolicC++ which defines the function `zero(x)` and `one(x)` which return zero and one in the same data type as `x`.

```
// gauss0power.cpp
// Gauss elimination + back substitution

#include <iostream>
#include <cmath>
#include "verylong.h"
#include "rational.h"
#include "identity.h"
using namespace std;

template<class T> T ABS(T& n)
{
  if(n < zero(T())) return -n;
  return n;
}

template<class T,int N> void gauss(T C[N][N+1])
{
  int i, j, k, m;
  for(i=0;i<N-1;i++)
  {
  T temp, max = ABS<T>(C[m=i][i]);
  for(j=i+1;j<N;j++)
   if(ABS<T>(C[j][i]) > max) max = ABS<T>(C[m=j][i]);
  for(j=i;j<=N;j++)
  { temp = C[i][j]; C[i][j] = C[m][j]; C[m][j] = temp; }
  if(C[i][i] != zero(T()))
   for(j=i+1;j<N;j++)
   {
    T r = C[j][i]/C[i][i];
    C[j][i] = zero(T());
    for(k=i+1;k<=N;k++) C[j][k] = C[j][k]-r*C[i][k];
   }
 }
}

template<class T,int N> int solve(T A[N][N],T b[N],T x[N])
{
  int i, j;
  T C[N][N+1];
  for(i=0;i<N;i++)
  {
  for(j=0;j<N;j++) C[i][j] = A[i][j];
  C[i][N] = b[i];
  }
  gauss<T,N>(C);         // Gauss elimination
  for(i=N-1;i>=0;i--)    // Back substitution
  {
```

```
  T sum = C[i][N];
  if(C[i][i] == zero(T())) return 0;
  for(j=i+1;j<N;j++) sum -= C[i][j]*x[j];
  x[i] = sum/C[i][i];
  }
  return 1;
}

int main(void)
{
  const int n = 3;
  double A[n][n] = {{ 1,  1,  1 },{ 8,  4,  2 },{ 27, 9, 3 }};
  double b[n] = { 1, 5, 14 };
  double x[n];
  if(solve<double,n>(A,b,x))
  {
  cout << "( ";
  for(int i=0;i<n;i++) cout << x[i] << " ";
  cout << ")" << endl;
  }
  else cout << "No solution or no unique solution" << endl;

  const int m = 2;
  Rational<Verylong> B[m][m] = {{ "1/2", "1/3" }, { "1/4", "1/5" }};
  Rational<Verylong> c[m] = { "1/6", "1/7" };
  Rational<Verylong> y[m];
  if(solve<Rational<Verylong>,m>(B,c,y))
  {
  cout << "( ";
  for(int i=0;i<m;i++) cout << y[i] << " ";
  cout << ")" << endl;
  }
  else cout << "No solution or no unique solution" << endl;
  return 0;
}
```

Problem 5. Consider the system of nonlinear equations

$$3x^2 - 2y^2 - 4z^2 + 54 = 0, \qquad 5x^2 - 3y^2 - 7z^2 + 74 = 0.$$

Write a C++ program that finds all integer solutions in the range $0 \le x \le 100$, $0 \le y \le 100$, $0 \le z \le 100$.

Solution 5. Using three `for`-loops we have the implementation

```
// nonlinear.cpp

#include <iostream>
```

```
using namespace std;

int main(void)
{
 for(int x=0;x<=100;x++)
 {
 for(int y=0;y<=100;y++)
 {
 for(int z=0;z<=100;z++)
 {
 int r1 = 3*x*x-2*y*y-4*z*z+54;
 int r2 = 5*x*x-3*y*y-7*z*z+74;
 if((r1==0) && (r2==0))
 {
 cout << "(" << x << "," << y << "," << z << ")"; cout << endl;
 }
 }
 }
 }
 return 0;
}
```

Problem 6. Let $a_0, a_1, a_2, \ldots, a_{n-1}$ be a finite sequence of numbers. Its *Cesáro sum* is defined as

$$\frac{1}{n}(s_0 + s_1 + s_2 + \cdots + s_{n-1})$$

where

$$s_k = a_0 + a_1 + \cdots + a_k$$

for each k, $0 \leq k \leq (n-1)$. Write a C++ program that finds the Cesáro sum for a given finite sequence of numbers. Use templates so that the program can also be used for complex numbers, rational numbers etc.

Solution 6. Using templates the program is

```
// cesarosums.cpp

#include <iostream>
#include <complex>
using namespace std;

template <class T> T cesaro(T* a,int n)
{
  T* s = new T[n];
  for(int j=0;j<n;j++) { s[j] = T(0); }
  for(int k=0;k<n;k++)
```

```
  { for(int l=0;l<=k;l++) { s[k] += a[l]; } }
  T CS = T(0);
  for(int p=0;p<n;p++) { CS += s[p]; }
  return CS/((T) n);
} // end function cesaro

int main(void)
{
 int n = 4;
 double* a = new double[n];
 a[0] = 0.5; a[1] = 0.7; a[2] = 0.9; a[3] = 0.3;
 double result;
 result = cesaro(a,n);
 cout << "result = " << result << endl;
 delete[] a;

 complex<double>* ac = new complex<double>[n];
 ac[0] = complex<double>(0.5,0.3);
 ac[1] = complex<double>(0.1,0.7);
 ac[2] = complex<double>(1.1,2.2);
 ac[3] = complex<double>(4.1,7.1);
 complex<double> sum;
 sum = cesaro(ac,n);
 cout << "sum = " << sum << endl;
 delete[] ac;
 return 0;
}
```

Problem 7. Let $x, y \in \mathbb{Z}$. Consider the equation

$$x^4 + y^4 + 79 = 48xy.$$

Write a C++ program that finds all solutions in the range $-10 \leq x \leq 10$ and $-10 \leq y \leq 10$.

Solution 7. If (x, y) is a solution, so are (y, x), $(-x, -y)$, and $(-y, -x)$. Obviously, $xy > 0$ so that x and y are both positive or negative. If $x = y$ there is no solution for $x^4 - 24x^2 + 79/2 = 0$. Suppose that (x, y) is a solution with $0 < y \leq x$. Then

$$x^3 + \frac{y^4 + 79}{x} = 48y \leq 48x.$$

It follows that $x^2 < 48$. Thus $|x|$ and $|y|$ are bounded by 6. Thus in our C++ program we can restrict us to this range.

Problem 8. Write a C++ program that finds the numbers of integer solutions of the equation $i_1 + i_2 + i_3 = 12$ satisfying the following constraints

$$0 \le i_1 \le 6, \quad 0 \le i_2 \le 6, \quad 0 \le i_3 \le 3.$$

Solution 8. A brute-force C++ program would be

```
// integersolutions.cpp
#include <iostream>
using namespace std;

int main(void)
{
  int i1, i2, i3;
  int count = 0;
  for(i1=0;i1<=6;i1++)
  {
  for(i2=0;i2<=6;i2++)
  {
  for(i3=0;i3<=3;i3++) { if((i1+i2+i3)==12) count++; }
  }
  }
  cout << "count = " << count << endl;
  return 0;
}
```

The number of solutions is 10.

Problem 9. Let x be the number of man, y be the number of woman and z be the number of children. Altogether there are 100 persons. Given 100 kg of potatos. Every man gets 3 portions, a woman gets 2 portions and a child gets 1/2 a portions. Find all (integer) solutions. We have two linear equations with three unkowns, however we have the constraint that x, y, z are nonnegative integers. Write a C++ program that finds all these integer solutions.

Solution 9. From

$$x + y + z = 100, \qquad 3x + 2y + \frac{1}{2}z = 100$$

we find after eliminating z

$$5x + 3y = 100.$$

The C++ program is

```
// Karl.cpp

#include <iostream>
```

```
using namespace std;

int main(void)
{
 int result;

 for(int x=0;x<100;x++)
 {
 for(int y=0;y<100;y++)
 {
 result = 100 - 5*x - 3*y;
 if(result==0)
 cout << "x = " << x << " " << "y = " << y << " "
      << "z = " << (100-x-y) << endl;
 }
 }
 return 0;
}
```

The seven solutions are

```
x =  2  y = 30  z = 68
x =  5  y = 25  z = 70
x =  8  y = 20  z = 72
x = 11  y = 15  z = 74
x = 14  y = 10  z = 76
x = 17  y =  5  z = 78
x = 20  y =  0  z = 80
```

Problem 10. Consider a linked list. Determine if the linked list loops using only two pointers.

Solution 10. We set both pointers to the head of the list.

Problem 11. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Consider the program (page 25) `au.cpp`. Where is the amplitude? Extend the program to two or more sine waves (for example frequency 880 besides 440).

Solution 11.

Problem 12. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Consider the program (page 8) `SineSound.java`. Run the Java program with the first line (page 10) changed to

```
...new File("sine.au"));
```

with `WAVE` also changed. Compare to the previous problem. Extend the program to get in more frequencies.

Solution 12.

Problem 13. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Consider the program (page 31) `LGB.java`. Rewrite the program in C++ either with the `vector` class of STL or "plain" (just functions).

Solution 13.

Problem 14. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Consider the program (page 44) `Noise.java`. Rewrite the program into C++.

Solution 14.

Problem 15. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Consider the Matlab Filter Implementation (page 49). Rewrite the code in C++.

Solution 15.

Problem 16. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Consider the program (page 60) `NonCircular.java`. Rewrite the code in C++.

Solution 16.

Problem 17. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Give a Java implementation of the two-dimensional convolution (page 61).

Solution 17.

Problem 18. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Implement the two-dimensional Fourier transform in C++ (page 72).

Solution 18.

Problem 19. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Implement the two-dimensional Cosine transform (page 76).

Solution 19.

Problem 20. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Use the `complex` class of STL and do someting useful with the z-transform (chapter 8).

Solution 20.

Problem 21. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Implement two-dimensional wavelets (page 89).

Solution 21.

Problem 22. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". In the C++ program (page 138) the total number of distinct observations is 2. Extend the program to more observations.

Solution 22.

Problem 23. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Extend the C++ code fragment (page 205) to a complete C++ program.

Solution 23.

Problem 24. The problem refers to the book "Mathematical Tools in Signal Processing with C++ and Java Simulations". Implement the decompression procedure (page 216) in C++.

Solution 24.

Problem 25. Write a Java program `Gauss.java` that implements Gauss elimination to solve linear equation with n equations and n unkowns. Apply the program to the system

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Solution 25. We implement the Gauss elimination with pivot search

```
// Gauss.java

public class Gauss
{
public static void solveGauss(double[][] A,double[] b,double[] x)
{
  int n = A.length;
  for(int j=0;j<n-1;j++)
  {
  double max = Math.abs(A[j][j]);
  int imax = j;
  for(int i=j+1;i<n;i++)
  {
  if(Math.abs(A[i][j]) > max)
  { max = Math.abs(A[i][j]);   imax = i; }
  }
  //
  double[] h = A[j]; A[j] = A[imax]; A[imax] = h;
  double hb = b[j]; b[j] = b[imax]; b[imax] = hb;

  for(int i=j+1;i<n;i++)
  {
  double f = -A[i][j]/A[j][j];
  for(int k=j+1;k<n;k++) A[i][k] += f*A[j][k];
  b[i] += f*b[j];
  }
 }
 for(int j=n-1;j>=0;j--)
 {
  x[j] = b[j];
  for(int k=j+1;k<n;k++) x[j] -= A[j][k]*x[k];
  x[j] /= A[j][j];
  }
  } // end method solveGauss

  public static void main(String[] args)
  {
  int n=3;
  double[][] A = new double[n][n];
  A[0][0] = 0.0; A[0][1] = 0.0; A[0][2] = 1.0;
  A[1][0] = 0.0; A[1][1] = 1.0; A[1][2] = 0.0;
  A[2][0] = 1.0; A[2][1] = 0.0; A[2][2] = 0.0;
  double[] b = new double[n];
  b[0] = 1.0; b[1] = 2.0; b[2] = 3.0;
  double[] x = new double[n];
  solveGauss(A,b,x);
  for(int j=0;j<n;j++)
```

```
  { System.out.println("x[" + j + "]= " + x[j]); }
  } // end main
} // end class Gauss
```

Problem 26. Which of the following C++ program fragments will loop forever?

```
int i = 1;
  while(i != 0) i = i + 1;

int j = 1;
  while(j != 0) j = 2*j + 1;

double x = 1.0;
  while(x != 0) x = x/2.0;
```

Solution 26.

Problem 27. Write down the tree expression for

$$((a/b) * 4) + ((1 + c) * (2 - d)).$$

Then evaluate the tree expression for $a = 1$, $b = 2$, $c = 3$, $d = 4$.

Solution 27.

Problem 28. How would one store a matrix using a linked list?

Solution 28.

Chapter 13

Applications of STL in C++

Problem 1. Given a set of positive integers, for example,

```
{ 2, 3, 11, 7, 4, 28, 41, 16 }
```

Write a C++ program using `set<int>` that partitions the set into two subsets of even and odd numbers, respectively.

Solution 1.

```
// partition.cpp

#include <iostream>
#include <set>
#include <utility>
using namespace std;

template <class T>
pair<set<T>,set<T> > partition(const set<T> &s,int (*p)(T))
{
  set<T> p1, p2;
  typename set<T>::const_iterator i;
  for(i=s.begin();i!=s.end();i++)
    if(p(*i)) p1.insert(*i); else p2.insert(*i);
  return make_pair(p1,p2);
}
```

```
int even(int x) { return x%2 == 0; }

template<class T>
void showset(const set<T> &s)
{
 typename set<T>::const_iterator i;

 cout << "{ ";
 for(i=s.begin();i!=s.end();)
 {
 cout << (*i);
 cout << ((++i==s.end()) ? " ":", ");
 }
 cout << "}";
}

int main(void)
{
  set<int> s;
  s.insert(2);  s.insert(3);
  s.insert(11); s.insert(7);
  s.insert(4);  s.insert(28);
  s.insert(41); s.insert(16);
  showset(s); cout << endl;
  pair<set<int>,set<int> > p = partition(s,even);
  showset(p.first); cout << endl;
  showset(p.second); cout << endl;
  return 0;
}
/*
{ 2, 3, 4, 7, 11, 16, 28, 41 }
{ 2, 4, 16, 28 }
{ 3, 7, 11, 41 }
*/
```

Problem 2. Write a useful C++ program that uses the function `find_if()`. The function `find_if` takes a predicate (function object or function) as parameter.

Solution 2.

```
// myfind_if.cpp

#include <iostream>
#include <list>
#include <string>
#include <algorithm>
using namespace std;
```

```
bool favorite_fruits(const std::string& name)
{ return (name == "apple" || name == "orange"); }

int main(void)
{
  list<string> fruits;
  fruits.push_back("banana");
  fruits.push_back("orange");
  fruits.push_back("peach");
  list<string>::const_iterator found =
    find_if(fruits.begin(),fruits.end(),favorite_fruits);
    if(found == fruits.end())
    cout << "No favorite fruits in the list";
    else cout << "Found it: " << *found << "\n";
  return 0;
}
```

Problem 3. A function f is a set of pairs (a, b) where $a \in A$ and $b \in B$ are elements of the sets A and B, respectively, such that for each $a \in A$ there is exactly one $b \in B$ such that $(a, b) \in f$.

Example. Consider the sets

$$A = \{one,\, two,\, three,\, four,\, five\}, \qquad B = \{1,\, 2,\, 3,\, 4,\, 5\}.$$

Let

$$f = \{(one, 1),\, (two, 2),\, (three, 3),\, (four, 4),\, (five, 5)\}$$

then $f(one) = 1$, $f(two) = 2$ etc. The set f denotes a function since each element of A appears as the first of a pair exactly once in f. The set A is called the domain of f and the set B is called the range of f.

We usually write $f : A \to B$ and $f(a) = b$ where $(a, b) \in f$.

Example. Consider

$$A = \{1,\, 2,\, 3,\, 4,\, 5\}, \quad B = \{true,\, false\}.$$

Let

$$g = \{(1, false),\, (2, true),\, (3, true),\, (4, false),\, (5, true)\}$$

then $g(2) = true$, $g(4) = false$ etc. The function g associates with each number in A the value in B which indicates whether the number is prime.

Let $f : A \to B$ and $g : B \to C$ where A, B and C are sets. Then, by convention, $a \in A \Rightarrow f(a) \in B$. Consequently

$$a \in A \Rightarrow f(a) \in B \Rightarrow g(f(a)) \in C.$$

Thus we define function composition

$$g \circ f : A \to C, \quad g \circ f = \{(a, g(f(a))) \,|\, a \in A\}.$$

Note that the functions f and g need to be compatible for function composition, i.e. the range of f must be contained in (subset of) the domain of g.

Example. Let f and g denote the two functions from the examples above and (we rename the sets for clarity)

$$\begin{aligned} A &= \{one,\, two,\, three,\, four,\, five\}, \\ B &= \{1,\, 2,\, 3,\, 4,\, 5\}, \\ C &= \{true,\, false\}. \end{aligned}$$

Thus $f : A \to B$ and $g : B \to C$. Consequently

$$\begin{aligned} g \circ f = \{&(one, g(f(one))),\, (two, g(f(two))),\, (three, g(f(three))), \\ &(four, g(f(four))),\, (five, g(f(five)))\} \\ f = \{&(one, g(1)),\, (two, g(2)),\, (three, g(3)),\, (four, g(4)),\, (five, g(5))\} \\ f = \{&(one, false),\, (two, true),\, (three, true),\, (four, false),\, (five, true)\}. \end{aligned}$$

In other words $(g \circ f)(two) = g(f(two)) = g(2) = true$ and $(g \circ f)(four) = false$.

Finite sets of a homogeneous nature, such as integers (represented by `int` in C++) allow a simple representation of functions in C++ using the STL data type `map`. The template class `map<A,B>` associates pairs of type `A` and `B`. Implement the examples above using the `map` data type. Implement function composition for two arbitrary (compatible) `map`s.

Solution 3.

```
// function.cpp

#include <iostream>
#include <map>
#include <set>
#include <string>
using namespace std;

template <class D,class R1,class R2>
map<D,R2> compose(map<R1,R2> m1,map<D,R1> m2)
{
 typename map<D,R1>::iterator i;
 map<D,R2> c;
 for(i=m2.begin();i!=m2.end();i++) c[i->first] = m1[i->second];
 return c;
```

```
}

template <class D,class R> set<D> domain(map<D,R> m)
{
 typename map<D,R>::iterator i;
 set<D> s;
 for(i=m.begin();i!=m.end();i++) s.insert(i->first);
 return s;
}

template <class D,class R> set<R> range(map<D,R> m)
{
 typename map<D,R>::iterator i;
 set<R> s;
 for(i=m.begin();i!=m.end();i++) s.insert(i->second);
 return s;
}

template<class T> void showset(const set<T> &s)
{
 typename set<T>::const_iterator i;
 cout << "{ ";
 for(i=s.begin();i!=s.end();)
 {
 cout << (*i);
 cout << ((++i==s.end()) ? " ":", ");
 }
 cout << "}";
}

int main(void)
{
 map<string,int> m;
 map<int,bool> prime1;
 m["zero"]  = 0; m["one"]   = 1; m["two"]   = 2;
 m["three"] = 3; m["four"]  = 4; m["five"]  = 5;
 prime1[0] = false; prime1[1] = false;
 prime1[2] = true;  prime1[3] = true;
 prime1[4] = false; prime1[5] = true;
 showset(domain(prime1)); cout << endl;
 showset(range(prime1));  cout << endl;
 map<string,bool> prime2 = compose(prime1,m);
 showset(domain(prime2)); cout << endl;
 showset(range(prime2));  cout << endl;
 map<string,bool>::iterator j;
 for(j=prime2.begin();j!=prime2.end();j++)
  cout << j->first << " -> " << j->second << endl;
 string n;
```

```
 cout << "Enter zero, one, two, three, four, or five: ";
 cin >> n;
 cout << n << ( (prime2[n]) ? " is ":" is not ") << "prime." << endl;
 return 0;
}
/*
{ 0, 1, 2, 3, 4, 5 }
{ 0, 1 }
{ five, four, one, three, two, zero }
{ 0, 1 }
five -> 1
four -> 0
one -> 0
three -> 1
two -> 1
zero -> 0
Enter zero, one, two, three, four, or five: four
four is not prime.
*/
```

Problem 4. The spiral map (Problems and Solutions in Scientific Computing, page 79) is a 1 to 1 map (i.e. invertible) which maps $\mathbb{Z}^2 \to \mathbb{Z}$. Write a C++ program using the `map` class and `pair` that stores the elements of the map as

```
map<int,pair<int,int> >
```

Solution 4.

```
// spiralmap2.cpp

#include <iostream>
#include <map>
#include <utility>
using namespace std;

template <class T1,class T2>
map<T2,T1> inverse(const map<T1,T2> &m)
{
 typename map<T1,T2>::const_iterator i;
 map<T2,T1> mi;
 for(i=m.begin();i!=m.end();i++) mi[i->second] = i->first;
 return mi;
}

int main(void)
{
 map<int,pair<int,int> > m;
```

```
 int max = 1000, x = 0, y = 0, i = 0;
 for(i=0;i<max;i++)
 {
 m[i] = make_pair(x,y);
 if((y>=x) && (y>-x))        --x;
 else if((y>x) && (y<=-x))   --y;
 else if((y<x) && (y>=-x+1)) ++y;
 else                        ++x;
 }
 cout << "Enter the number: "; cin >> i;
 cout << i << " -> (" << m[i].first << ","
                      << m[i].second << ")" << endl;
 map<pair<int,int>,int> mi = inverse(m);
 cout << "Enter x: "; cin >> x;
 cout << "Enter y: "; cin >> y;
 cout << "(" << x << "," << y << ")" << " -> "
      << mi[make_pair(x,y)] << endl;
  return 0;
}
```

Problem 5. Let $\mathbb{C}$ be the complex plane. Let $c \in \mathbb{C}$. The *Mandelbrot set* M is defined as

$$M := \{\, c \in \mathbb{C} : c,\, c^2 + c,\, (c^2 + c)^2 + c, \ldots, \not\to \infty \,\}.$$

To find the Mandelbrot set we study the recursion relation

$$z_{t+1} = z_t^2 + c, \qquad t = 0, 1, 2, \ldots$$

with the initial value $z_0 = 0$ and whether z_t escapes to infity. For example $c = 0$ and $c = 1/4 + i/4$ belong to the Mandelbrot set. The point $c = 1/2$ does not belong to the Mandelbrot set. Write a C++ program using the complex class of STL to find the Mandelbrot set. The output should be written to a file `Mandel.pnm` (portable anymap utilities). This file can then be used to display the fractal.

Solution 5. Using `complex<double> z` we have the implementation

```
// mandelbrot.cpp

#include <complex>
#include <fstream>
#include <iostream>
using namespace std;

int mandeltest(const complex<double> &c,int maxiter)
{
```

```
  int j;
  complex<double> z;
  for(j=0,z=0.0;j<maxiter;j++)
  {
  z=z*z+c;
  if(abs(z)>100) return j/int(1.0+exp(-abs(z)));
  }
  return j;
}

int main(void)
{
  int maxiter=255, dpi=180, xcm=100, ycm=200;
  int i, j;
  double minx=-1.33, maxx=-1.25, miny=-0.20, maxy;
  double x, y, stepx, stepy;
  int xcount=int(dpi*xcm/2.5), ycount=int(dpi*ycm/2.5);
  ofstream mandel("mandel.pnm");
  maxy=(maxx-minx)*ycount/xcount+miny;
  stepx=(maxx-minx)/xcount;
  stepy=(maxy-miny)/ycount;
  cout << xcount << " x " << ycount << endl;
  mandel << "P6 " << xcount << " "
         << ycount << " " << maxiter << endl;
  for(i=0,y=miny;i<ycount;y+=stepy,i++)
  {
  cout.precision(3);
  cout << 100.0*i/ycount << "% \r";
  cout.flush();
  for(j=0,x=minx;j<xcount;x+=stepx,j++)
  {
  int k=mandeltest(complex<double>(x,y),maxiter);
  unsigned char c;
  c=30+char(205*((x-minx)*k/((maxx-minx)*maxiter)));
  mandel.write((char*)&c,1);
  c=char(255*((y-miny)*k/((maxy-miny)*maxiter)));
  c=20;
  mandel.write((char*)&c,1);
  c=30+char(225*double(k)/maxiter);
  mandel.write((char*)&c,1);
  }
}
  mandel.close();
  cout << endl;
  return 0;
}
```

Problem 6. A priority queue is a type of queue that assigns a priority to every element that it stores. New elements are added to the queue using the `push()` function. Thus it is a set for which two operations are defined:
1) Adding an item (using `push()`)
2) Extracting the item that has the highest priority using `top()` and `pop()`.

We may think of a priority queue as a set of tasks with priorities. At any time a new task can be added. A task can also be removed from the priority queue, but this can only be the one with the highest priority. If this highest priority is shared by more than one task, we do not care which one is taken. Write a C++ program that uses the priority queue from STL. Apply it to floating point numbers so that bigger numbers get a higher priority. Apply it to strings so that strings lexicographicaly higher get a higher priority (case sensitive).

Solution 6. As container we use the `vector` class.

```
// Priorityqueue.cpp

#include <iostream>
#include <queue>
#include <vector>
#include <string>
#include <cassert>
using namespace std;

int main(void)
{
  priority_queue<double,vector<double>,less<double> > q;
  assert(q.empty());
  q.push(3.1356); q.push(4.134); q.push(2.1);
  for(int j=0;j<3;j++)
  {
  cout << q.top() << " ";
  q.pop();
  }
  priority_queue<string,vector<string>,less<string> > p;
  assert(p.empty());
  p.push("willi");
  p.push("xxx");
  p.push("ccaa");
  for(int k=0;k<3;k++) { cout << p.top() << " "; p.pop(); }
  return 0;
}
```

Problem 7. The ancient puzzle of the Tower of Hanoi consists of a number of wooden disks mounted on three poles, which are in turn attached to a baseboard.

The disks each have different diameters and a hole in the middle large enough for the poles to pass through. At the beginning all disks are on the left pole with the smallest at the top, the second smallest one down etc. The object of the puzzle is to move all the disks over to the right pole, one at the time, so that they end up in the original order on that pole. One uses the middle pole as a temporary resting place for the disks. However it is allowed for a larger disk to be on top of a smaller one. For example if we have three disks then the moves are

```
move disk A from pole 1 to 3
move disk B from pole 1 to 2
move disk A from pole 3 to 2
move disk C from pole 1 to 3
move disk A from pole 2 to 1
move disk B from pole 2 to 3
move disk A from pole 1 to 3
total number of moves: 7
```

(i) Write a C++ program using recursion to implement the Tower of Hanoi.
(ii) Write a C++ program using the `stack` class of the standard template libraray to implement the Tower of Hanoi.

Solution 7. The solution is

```
// hanoist.cpp

#include <iostream>
#include <stack>
using namespace std;

void hanoi(int n,char A,char B,char C)
{
 if(n == 0) return;
 hanoi(n-1,A,C,B);
 cout << "Move " << A << " to " << C << endl;
 hanoi(n-1,B,A,C);
}

struct hanoi_struct { int n; char A, B, C; };

void hanoist(int n,char A,char B,char C)
{
 struct hanoi_struct hs, hss;
 stack<hanoi_struct> st;
 hs.n = n; hs.A = A; hs.B = B; hs.C = C;
 st.push(hs);
 while(!st.empty())
 {
  hs = st.top(); st.pop();
```

```
  switch(hs.n)
  {
   case 0:  break;
   case 1:  cout << "Move " << hs.A << " to " << hs.C << endl;
            break;
   default: // operations in reverse order
            hss.n = hs.n-1; hss.A = hs.B; hss.B = hs.A; hss.C = hs.C;
            st.push(hss);
            hss.n = 1; hss.A = hs.A; hss.B = hs.B; hss.C = hs.C;
            st.push(hss);
            hss.n = hs.n-1; hss.A = hs.A; hss.B = hs.C; hss.C = hs.B;
            st.push(hss);
  }
 }
}

int main(void)
{
 hanoi(4,'A','B','C'); cout << endl;
 hanoist(4,'A','B','C');
 return 0;
}
```

Problem 8. Using the `Verylong` class of SymbolicC++ and the `complex` class (of STL) that finds positive integer solutions (a, b, c) of the equation

$$c = (a + bi)^3 - 107i$$

where $i^2 = -1$.

Solution 8.

Problem 9. Let $i = \sqrt{-1}$. Calculate i^i. Use the `complex` class of the standard template library of C++ to calculate i^i. Discuss.

Solution 9. The program is

```
// ipoweri.cpp

#include <iostream>
#include <complex>
using namespace std;

int main(void)
{
 complex<double> i(0.0,1.0);
 complex<double> r = pow(i,i);
```

```
 cout << "r = " << r;
 return 0;
}
```

Chapter 14

Particle Swarm Optimization

Problem 1. Particle Swarm Optimization (PSO) is based on the behavior of a colony or swarm of insects, such as ants, termites, bees, and wasps; a flock of birds; or a school of fish. The particle swarm optimization algorithm mimics the behavior of these social organisms. The word particle denotes, for example, a bee in a colony or a bird in a flock. Each individual or particle in a swarm behaves in a distributed way using its own intelligence and the collective or group intelligence of the swarm. As such, if one particle discovers a good path to food, the rest of the swarm will also be able to follow the good path instantly even if their location is far away in the swarm. Optimization methods based on swarm intelligence are called behaviorally inspired algorithms as opposed to the genetic algorithms, which are called evolution-based procedures. The PSO algorithm was originally proposed by Kennedy and Eberhart in 1995.

In the context of multivariable optimization, the swarm is assumed to be of specified or fixed size with each particle located initially at random locations in the multidimensional design space. Each particle is assumed to have two characteristics: a position and a velocity. Each particle wanders around in the design space and remembers the best position (objective function value) it has discovered. The particles communicate information or good positions to each other and adjust their individual positions and velocities based on the information received on the good positions.
The PSO is developed based on the following model:

1. When one particle locates an extremum point of the objective function, it

instantaneously transmits the information to all other particules.
2. All other particles gravitate to the extremum point of the objective function,

but not directly.
3. There is a component of each particle's own independent thinking as well as

its past memory.
Thus the model simulates a random search in the design space for the extremum

points of the objective function. As such, gradually over many iterations, the particles go to the target (the extremum point of the objective function).
The algorithm for determining the maximum of a function $f(\mathbf{x})$ (with $\mathbf{x}$ an n

dimensional vector) is as follows:

1) Initialize the number of particles N, the search intervals for each dimension (a_i, b_i), $i = 1, \ldots, n$ (n being the dimension of the search space), the search precision for each dimension ϵ_i, the maximum number of iterations i_{max}.
Initialize the positions of the particles $\mathbf{x}_j(0) = rand()$, $j = 1, \ldots, N$ randomly in the search domain.
Initialize the speeds of the particles $\mathbf{v}_j(0) = 0$, $j = 1, \ldots, N$.
Initialize the individual best positions of the particles $\mathbf{x}_{best,j}(0) = \mathbf{x}_j(0)$, $j = 1, \ldots, N$.
Initialize the iteration count $k = 0$.

2) Check the stop conditions: the diameter of the swarm in each dimension is less then the dimension's precision ϵ_i, or the maximum number of iterations i_{max} was reached. If yes then terminate else continue to step 3).

3) Calculate the values of the function $f(\mathbf{x})$ in the current positions of the particles, $f(\mathbf{x}_j(k))$, $j = 1, \ldots, N$. Update the values of the best individual points for each particle $\mathbf{x}_{best,j}(k)$, $j = 1, \ldots, N$ and the value of the global best point $\mathbf{x}_{best}(k)$. Continue to 4).

4) Update the particles' speeds by applying the formula:

$$\mathbf{v}_j(k) = \theta(k)\mathbf{v}_j(k-1) + c_1 r_1 \left[\mathbf{x}_{best,j}(k) - \mathbf{x}_j(k-1)\right] + c_2 r_2 \left[\mathbf{x}_{best}(k) - \mathbf{x}_j(k-1)\right]$$

where $j = 1, \ldots, N$ and r_1 and r_2 are random number between 0 and 1. The parameters c_1 and c_2 have usually the value 2 so that the particles would overfly the target about half of the time. $\theta(k)$ is the inertia weight dependent of the iteration count according to the formula:

$$\theta(k) = \theta_{max} - \left(\frac{\theta_{max} - \theta_{min}}{i_{max}}\right) k$$

where θ_{max} is a maximum value and θ_{min} is a minimum value, typically $\theta_{max} = 0.9$ and $\theta_{min} = 0.4$. Continue to 5).
5) Update the particles' positions by applying the formula:

$$\mathbf{x}_j(k) = \mathbf{x}_j(k-1) + \mathbf{v}_j(k), \quad j = 1, \ldots, N$$

6) Increment the iteration count k and go to 2).

Determine the maximum of the function

$$f(x) = -x^2 + 2x + 11, \ -2 \leq x \leq 2$$

by applying the PSO (Particle Swarm Optimization) method. The required precision is 10^{-4}.

Solution 1. The following C# program gives a solution implementation.

```
// PSO.cs
using System;
using System.Diagnostics;

namespace Optimization
{
 // Particle Swarm Optimization
 class PSO
 {
 public delegate double Func(double x);

 public PSO(Func func,double a,double b,double eps,int N,int maxiter)
 {
 _func = func;
 _a = a; _b = b; _eps = eps;
 _N = N;
 _maxiter = maxiter;
 _dtheta = (_thetamax-_thetamin)/_maxiter;
 }

  public double Xbest { get; set; }
  public double Fbest { get; set; }
  public int Iter { get; set; }

  public void Calculate()
  {
  double[] x = new double[_N];
  double[] v = new double[_N];
  double[] best = new double[_N];
```

```
double[] fbest = new double[_N];
double f, min, max;
int i;
// Initialize the positions randomly in (_a, _b)
Random rnd = new Random();
double delta = _b - _a;
for(i=0;i < _N;i++)
{ x[i] = _a + rnd.NextDouble() * delta; }
// Initialize the best
for(i=0;i < _N;i++)
{
best[i] = x[i];
fbest[i] = _func(x[i]);
}
// Initialize the velocities
for(i=0;i < _N;i++)
{ v[i] = 0.0; }
Xbest = 0.0;
Fbest = -Double.MaxValue;
double D = 2.0*_eps, theta = _thetamax;
Iter = 0;
while((D > _eps) && (Iter <= _maxiter))
{
for(i=0;i < _N;i++)
{
f = _func(x[i]);
if(f > fbest[i]) { best[i] = x[i]; fbest[i] = f; }
if(f > Fbest) { Xbest = x[i]; Fbest = f; }
}
for(i=0;i<_N;i++)
{
v[i] = theta*v[i] + _c1*rnd.NextDouble()*(best[i] - x[i]) +
      _c2 * rnd.NextDouble() * (Xbest - x[i]);
x[i] += v[i];
// Check the ranges
if(x[i] < _a) { x[i] = _a; }
if(x[i] > _b) { x[i] = _b; }
}
// Calculate the new diameter
min = Double.MaxValue; max = -Double.MaxValue;
for(i=0;i < _N;i++)
{
if(x[i] > max) max = x[i];
if(x[i] < min) min = x[i];
}
D = max - min;
theta -= _dtheta;
Iter++;
```

```
  }
  }

  public void Print()
  {
  Console.WriteLine("Xbest = " + Xbest);
  Console.WriteLine("Fbest = " + Fbest);
  Console.WriteLine("Iter = " + Iter);
  }

  private static double _c1 = 2.0;
  private static double _c2 = 2.0;
  private static double _thetamin = 0.1;
  private static double _thetamax = 0.9;
  private int _N;
  private int _maxiter;
  private double _dtheta;
  private double _a;
  private double _b;
  private double _eps;
  private Func _func;

  public static void Main(String[] args)
  {
  // anonymous function
  Func f = delegate(double x) { return -x*x + 2.0*x + 11.0; };
  double a = -2.0, b = 2.0;
  double eps = 1.0e-4;
  int N = 10;
  int maxiter = 200;
  PSO pso = new PSO(f, a, b, eps, N, maxiter);
  pso.Calculate();
  pso.Print();
  Console.WriteLine("Xbest = " + pso.Xbest);
  Console.ReadLine();
  }
  }
}
```

The results among runs vary due to the stochastic characteristics of the method. Typical output results:

```
Xbest = 0.999999972070712
Fbest = 12
Iter = 118
Xbest = 0.999999972070712
```

Problem 2. Minimize

$$f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 4x_2$$

subject to the constraints

$$x_1 + 4x_2 - 5 \leq 0, \quad 2x_1 + 3x_2 - 6 \leq 0, \quad x_1 \geq 0, \quad x_2 \geq 0$$

by applying the Particle Swarm Optimization (PSO) method. The required precision is 10^{-3}.

Solution 2. The theory of the PSO method was described in the previous problem. What is different in the case of a constrained optimization problem is the way in which the functional to be minimized is defined. If we consider the general constrained minimization problem:
Minimize

$$f(\mathbf{x})$$

subject to

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m$$

and

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, p,$$

we can reduce this problem to an unconstrained minimization problem by applying a penalty function method, for example the exterior method, which assumes the minimization of the functional:

$$\Phi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{n} r_i \tilde{g}_i(\mathbf{x})^2 + \sum_{j=1}^{p} R_j h_j(\mathbf{x})^2$$

where $\tilde{g}_i(\mathbf{x}) = \max\{g_i(\mathbf{x}), 0\}$ and r_i, R_j are some appropiate large positive constants.

The C# code is the same as for the previous problem. Only the `Main()` function is different.

```
public static void Main(String[] args)
{
 Func f = delegate(double x1,double x2)
 {
 double g1 = x1 + 4.0 * x2 - 5.0;
 if(g1 <= 0.0) g1 = 0.0;
 double g2 = 2.0 * x1 + 3.0 * x2 - 6.0;
 if(g2 <= 0.0) g2 = 0.0;
 return x1*x1 + x2*x2 - 2.0*x1 - 4.0*x2 + 100.0*g1*g1 + 100.0*g2*g2;
 };
```

```
 double a1 = 0.0, b1 = 5.0;
 double a2 = 0.0, b2 = 2.0;
 double eps = 1.0e-3;
 int N = 20;
 int maxiter = 300;
 PSO pso = new PSO(f, a1, b1, a2, b2, eps, N, maxiter);
 pso.Calculate();
 pso.Print();
 Console.ReadLine();
}
```

Considering the stochastic nature of the PSO method, the output results vary. Some typical results are:

```
Xbest = 0.764842337759194
Ybest = 1.05937731766136
Fbest = -4.05937683715063
Iter = 238

Xbest = 0.764844055622397
Ybest = 1.05937686791438
Fbest = -4.0593768371545
Iter = 260
```

Problem 3. Find the global minimum of the De Jong's function (or sphere model)

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}_i^2, \;\; n \geq 2$$

by applying the Particle Swarm Optimization (PSO) method. The required precision is 10^{-3}.

Solution 3. The theory of the PSO method was described above Here only the C# code is implemented in a more general manner.

```
// PSO.cs

using System;
using System.Collections.Generic;

namespace Optimization
{
  class Vector
  {
  //Constructors
  public Vector(int n)
  {
```

```
_v = new List<double>(n);
for(int i=0;i < n;i++) { _v.Add(0.0); }
}

public Vector(List<double> v)
{
_v = new List<double>(v);
}

public Vector(Vector v)
{
_v = new List<double>(v._v);
}

public int Size { get { return _v.Count; } }

// Indexer
public double this[int ix]
{
get { return _v[ix]; }
set { _v[ix] = value; }
}

public void Copy(Vector v)
{
for(int i=0;i < _v.Count;i++) { _v[i] = v[i]; }
}

// Minus
public static Vector operator -(Vector v)
{
Vector res = new Vector(v);
for(int i=0;i < res.Size;i++) { res[i] = -res[i]; }
return res;
}

// Product with a scalar
public static Vector operator *(double a, Vector v)
{
Vector res = new Vector(v);
for(int i=0;i < res.Size;i++) { res[i] *= a; }
return res;
}

// Sum
public static Vector operator +(Vector a, Vector b)
{
Vector res = new Vector(a);
```

```
for(int i=0;i < res.Size;i++) { res[i] += b[i]; }
return res;
}

// Difference
public static Vector operator -(Vector a, Vector b)
{
Vector res = new Vector(a);
for(int i=0;i < res.Size;i++) { res[i] -= b[i]; }
return res;
}

// Scalar Product
public static double operator *(Vector a, Vector b)
{
double prod = 0.0;
for(int i=0;i < a.Size;i++) { prod += a[i]*b[i]; }
return prod;
}

public double Norm1()
{
double sum = 0.0;
for(int i=0;i < _v.Count;i++) { sum += Math.Abs(_v[i]); }
return sum;
}

public double Norm2()
{
double sum = 0.0;
for(int i=0;i < _v.Count;i++) { sum += _v[i]*_v[i]; }
return Math.Sqrt(sum);
}

public void Normalize2()
{
double norm = this.Norm2();
for(int i=0;i < _v.Count;i++) { _v[i] /= norm; }
}

private List<double> _v = null;
}

// Particle Swarm Optimization
class PSO
{
public delegate double Func(Vector v);
```

```
public PSO(Func func,int n,double a,double b,double eps,int N,int maxiter)
{
_func = func;
_a = a; _b = b; _n = n;
_eps = eps;
_N = N;
_maxiter = maxiter;
_dtheta = (_thetamax - _thetamin)/_maxiter;
}

public Vector Xbest { get; set; }
public double Fbest { get; set; }
public int Iter { get; set; }

public void Calculate()
{
Vector[] x = new Vector[_N];
Vector[] v = new Vector[_N];
Vector[] best = new Vector[_N];
double[] fbest = new double[_N];
double f, min, max;
int i, j;
// Initialize the positions randomly
Random rnd = new Random();
double delta = _b - _a;
for(i=0;i < _N;i++)
{
x[i] = new Vector(_n);
for(j=0;j< _n;j++)
{
x[i][j] = _a + rnd.NextDouble()*delta;
}
}
// Initialize the best
for(i=0;i<_N;i++)
{
best[i] = new Vector(x[i]);
fbest[i] = _func(x[i]);
}
// Initialize the velocities
for(i=0;i<_N;i++)
{
v[i] = new Vector(_n);
}
Xbest = new Vector(_n);
Fbest = Double.MaxValue;
double D = 2.0*_eps, theta = _thetamax;
Vector dx = new Vector(_n);
```

```
Iter = 0;
while((D > _eps) && (Iter <= _maxiter))
{
for(i=0;i<_N;i++)
{
f = _func(x[i]);
if(f < fbest[i]) { best[i].Copy(x[i]); fbest[i] = f; }
if(f < Fbest)
{
Xbest.Copy(x[i]);
Fbest = f;
}
}
for(i=0;i < _N;i++)
{
for(j=0;j < _n;j++)
{
v[i][j] = theta*v[i][j]+ _c1*rnd.NextDouble()*(best[i][j] - x[i][j]) +
_c2 * rnd.NextDouble()*(Xbest[j]-x[i][j]);
x[i][j] += v[i][j];
// Check the ranges
if(x[i][j] < _a) { x[i][j] = _a; }
if(x[i][j] > _b) { x[i][j] = _b; }
}
}
// Calculate the new diameter
for(j=0;j<_n;j++)
{
min = Double.MaxValue;
max = -Double.MaxValue;
for(i=0;i<_N;i++)
{
if(x[i][j] > max) max = x[i][j];
if(x[i][j] < min) min = x[i][j];
}
dx[j] = max - min;
}
D = dx.Norm2();
theta -= _dtheta;
Iter++;
}
}

public void Print()
{
Console.Write("( ");
for(int j=0;j<_n;j++) { Console.Write(Xbest[j] + ","); }
Console.WriteLine(")");
```

```
  Console.WriteLine("Fbest = " + Fbest);
  Console.WriteLine("Iter = " + Iter);
  }

  private static double _c1 = 2.0;
  private static double _c2 = 2.0;
  private static double _thetamin = 0.1;
  private static double _thetamax = 0.9;
  private int _N; // swarm size
  private int _n; // dimensions
  private int _maxiter;
  private double _dtheta;
  private double _a;
  private double _b;
  private double _eps;
  private Func _func;

  public static void Main(String[] args)
  {
  // anonymous function
  Func f = delegate(Vector v)
  {
  double res = 0.0;
  for(int i=0;i < v.Size;i++) { res += v[i]*v[i]; }
  return res;
  };
  double a = -5.12, b = 5.12;
  double eps = 1.0e-3;
  int N = 30; // swarm size
  int n = 10; // dimensions
  int maxiter = 300;
  PSO pso = new PSO(f, n, a, b, eps, N, maxiter);
  pso.Calculate();
  pso.Print();
  Console.ReadLine();
  }
  }
}
```

Considering the stochastic nature of the PSO method, the output results vary. Some typical results are:

```
Xbest = ( 1.75743850620593E-06, -7.04582167753594E-07, -5.98369689492146E-07,
         -1.47577290957276E-07, 1.82807687033641E-06, -1.63155487700595E-06,
         -1.89989811929982E-06, -4.35238981607769E-07, 1.44554970651784E-06,
          1.32364807623722E-06, )
Fbest = 1.7609391855432E-11
Iter = 258

Xbest = ( 1.85150232714368E-06, 5.36097754838293E-06, -7.88392060112956E-06,
```

```
        -1.05494321354011E-06, 8.78338039213776E-06, -4.18653688743919E-06,
        -5.73282698952597E-06, 3.23069025424733E-06, 3.87355624888027E-07,
        -4.25035494605146E-06, )
Fbest = 2.51630338951555E-10
Iter = 247
```

Problem 4. Differntial evolution (DE) is a population-based optimization method that attacks the starting point problem by sampling the objective function at multiple, randomly chosen initial points. At initialization a vector population of dimension N_p is generated such that the allowed parameter region is entirely covered. Each vector is indexed with a number from 0 to $N_p - 1$ for bookkeeping because each of them has to enter a competition. Like other Evolutionary Strategy population-based methods, DE generates new points that are perturbations of existing points, but unlike other Evolutionary Strategy methods, DE perturbs population vectors with the scaled difference of two randomly selected population vectors to produce the trial vectors. Assume that we produce the trial vector with index 0, $\mathbf{u}_0$. DE selects randomly two distinct vectors $\mathbf{x}_{r1}$ and $\mathbf{x}_{r2}$ from the population and adds the scaled perturbation $\mathbf{x}_{r1} - \mathbf{x}_{r2}$ to a third vector also randomly selected from the population $\mathbf{x}_{r3}$, distinct from the first two vectors. The procedure is repeated in order to generate all the set of trial vectors $\mathbf{u}_0$, $\mathbf{u}_1$, ..., $\mathbf{u}_{N_p}$. In the selection stage, each trial vector competes against the vector in the population vectors with the same index. The vector with the lower objective function value is selected as a member of the next generation. The survivors of the N_p pairwise competitions become parents for the next generation in the evolutionary cycle. The evolutionary cycles are repeated until a termination criteria is meet, for example the diameter of the population becomes less than a small limit value ϵ or a maximum number of population generations were generared.
The more detailed algorithm for determining the global minimum of a function

$f(\mathbf{x})$ (with $\mathbf{x}$ an n dimensional vector) is as follows:
1) *Initialization* - Initialize the number of vectors N_p, the search intervals for

each dimension (a_i, b_i), $i = 1, \ldots, n$, the precision for the termination criteria ϵ, the maximum number of iterations i_{max}, the scale factor $F \in (0, 1+)$, the crossover probability $C_r \in [0, 1)$. Initialize the positions of the particles $\mathbf{x}_{i,j}^{(0)} = a_j + rand_{i,j}()(b_j - a_j)$, $i = 1, \ldots, N_p$ $j = 1, \ldots, n$ randomly in the search domain (the random number generator, $rand()$, returns a uniformly distributed random number from within the range $[0, 1)$). Initialize the iteration count $k = 0$.
2) *Mutation* - differential mutation adds a scaled, randomly sampled, vector

difference to a third randomly sampled vector to create a mutant vector

$$\mathbf{v}_i^{(k)} = \mathbf{x}_{r3}^{(k)} + F(\mathbf{x}_{r1}^{(k)} - \mathbf{x}_{r2}^{(k)}) \;\; i = 1, \ldots N_p$$

The scale factor, $F \in (0, 1+)$, is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values are seldom greater than 1. The base vector index, $r3$, can be determined in a variety of ways, but for now it is assumed to be a randomly chosen vector index that is different from the target vector index, i. Except for being distinct from each other and from both the base and target vector indices, the difference vector indices, $r1$ and $r2$, are also randomly selected once per mutant.

3) *Crossover* - to complement the differential mutation search strategy, DE also employs uniform crossover. Sometimes referred to as discrete recombination, (dual) crossover builds trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector:

$$\mathbf{u}_{i,j}^{(k)} = \begin{cases} \mathbf{v}_{i,j}^{(k)} & \text{if } rand()_{i,j} < C_r \text{ or } j = j_{rand} \\ \mathbf{x}_{i,j}^{(k)} & \text{otherwise} \end{cases} \quad i = 1, \ldots, N_p, \;\; j = 1, \ldots, n$$

The crossover probability, $C_r \in [0, 1)$, is a user-defined value that controls the fraction of parameter values that are copied from the mutant. To determine which source contributes a given parameter, uniform crossover compares C_r to the output of a uniform random number generator. If the random number is less than or equal to C_r, the trial vector component is inherited from the mutant $\mathbf{v}_i^{(k)}$; otherwise, the trial vector component is copied from the vector, $\mathbf{x}_i^{(k)}$. In addition, the trial vector component with randomly chosen index, j_{rand}, is taken from the mutant to ensure that the trial vector does not duplicate $\mathbf{x}_i^{(k)}$. Because of this additional demand, C_r only approximates the true probability, p_{C_r}, that a trial parameter will be inherited from the mutant.

4) *Selection* - if the trial vector, $\mathbf{u}_i^{(k)}$, has an equal or lower objective function value than that of its target vector, $\mathbf{x}_i^{(k)}$, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation:

$$\mathbf{x}_i^{(k)} = \begin{cases} \mathbf{u}_i^{(k)} & \text{if } f(\mathbf{u}_i^{(k)}) < f(\mathbf{x}_i^{(k)}) \\ \mathbf{x}_i^{(k)} & \text{otherwise} \end{cases} \quad i = 1, \ldots, N_p$$

By comparing each trial vector with the target vector from which it inherits parameters, DE more tightly integrates recombination and selection than do other Evolutionary Algorithms.

5) *Termination* - check the termination conditions: the diameter of the population is less then the preset precision ϵ, or the maximum preset number of generations i_{max} was reached. If yes then terminate, else increment the iteration count k and go to 2).

Determine the global minimum of the function

$$f(x,y) = 3(1-x)^2 e^{-(x^2+(y+1)^2)} - 10(\frac{x}{5} - x^3 - y^5)e^{-(x^2+y^2)} - \frac{1}{3}e^{-((x+1)^2+y^2)}$$

where

$$-4 \leq x \leq 4, \quad -4 \leq y \leq 4$$

by applying the DE method. The required precision is 10^{-4}.

Solution 4. The following C# program gives a solution implementation.

```
// DE.cs
using System;

namespace Optimization
{
  // Differential Evolution
  class DE
  {
  public delegate double Func(double x, double y);

  public DE(Func func,double a1,double b1,double a2,double b2,int N,
            double eps,double c,double f)
  {
  _func = func;
  _a1 = a1; _b1 = b1; _a2 = a2; _b2 = b2;
  _N = N;
  _eps = eps;
  _c = c;
  _f = f;
  }

  public double Xbest { get; set; }
  public double Ybest { get; set; }
  public double Fbest { get; set; }
  public int Iter { get; set; }

  public void Calculate()
  {
  if(_calculated)
  {
  // already calculated
  return;
  }
  _calculated = true;
  Random rnd = new Random();
```

```
int i;
int r0, r1, r2, jrand;
double diameter = 2.0 * _eps;
double xmin, xmax, ymin, ymax;
double diameterx = _b1 - _a1;
double diametery = _b2 - _a2;
// function values
double[] f = new double[_N];
double[] x = new double[_N];
double[] y = new double[_N];
double[] ux = new double[_N];
double[] uy = new double[_N];
for(i=0; i<_N;i++)
{
x[i] = _a1 + rnd.NextDouble() * diameterx;
y[i] = _a2 + rnd.NextDouble() * diametery;
f[i] = _func(x[i], y[i]);
}
double temp;
Iter = 0;
while(true)
{
for(i=0;i<_N;i++)
{
// r0 != r1 != r2 != i
do r0 = rnd.Next(_N); while (r0 == i);
do r1 = rnd.Next(_N); while ((r1==r0) || (r1==i));
do r2 = rnd.Next(_N); while ((r2==r1) || (r2==r0) || (r2==i));
jrand = rnd.Next(2);
if((rnd.NextDouble() < _c) || (jrand == 0))
{
ux[i] = x[r0] + _f * (x[r1] - x[r2]);
// Check the bounds
if(ux[i] < _a1) { ux[i] = _a1; }
if(ux[i] > _b1) { ux[i] = _b1; }
}
else { ux[i] = x[i]; }
if((rnd.NextDouble() < _c) || (jrand == 1))
{
uy[i] = y[r0] + _f*(y[r1]-y[r2]);
// Check the bounds
if(uy[i] < _a2) { uy[i] = _a2; }
if(uy[i] > _b2) { uy[i] = _b2; }
}
else { uy[i] = y[i]; }
}
// Select the next generation
for(i=0;i<_N;i++)
```

```
{
temp =  _func(ux[i], uy[i]);
if(temp < f[i])
{ x[i] = ux[i]; y[i] = uy[i]; f[i] = temp; }
}
Fbest = Double.MaxValue;
for(i=0; i<_N;i++)
{
if(f[i] < Fbest) { Fbest = f[i]; Xbest = x[i]; Ybest = y[i]; }
}
Iter++;
//The new diameters
xmin = Double.MaxValue;
xmax = -Double.MaxValue;
ymin = Double.MaxValue;
ymax = -Double.MaxValue;
for(i=0;i<_N;i++)
{
if(x[i] < xmin) { xmin = x[i]; }
if(x[i] > xmax) { xmax = x[i]; }
if(y[i] < ymin) { ymin = y[i]; }
if(y[i] > ymax) { ymax = y[i]; }
}
diameterx = xmax - xmin;
diametery = ymax - ymin;
diameter = Math.Sqrt(diameterx*diameterx + diametery*diametery);
if(diameter < _eps) // Stop condition
{ break; }
}
}

public void Print()
{
Console.WriteLine("Xbest = " + Xbest);
Console.WriteLine("Ybest = " + Ybest);
Console.WriteLine("Fbest = " + Fbest);
Console.WriteLine("Iter = " + Iter);
}

private int _N;
private double _a1; private double _b1;
private double _a2; private double _b2;
private double _eps;
private double _c;
private double _f;
private Func _func;
private bool _calculated = false;
```

```
  public static void Main(String[] args)
  {
  // anonymous function
  Func func = delegate(double x, double y)
  {
  double onemx = 1.0 - x;
  double xp1 = x + 1.0;
  double yp1 = y + 1.0;
  double x2 = x*x;
  double y2 = y*y;
  return 3.0*onemx*onemx*Math.Exp(-(x2+yp1*yp1)) -
         10.0*(x/5.0-x*x2-y2*y2*y)*Math.Exp(-(x2+y2)) -
         Math.Exp(-(xp1*xp1+y2))/3.0;
  };
  int N = 30;
  double a1 = -4.0, b1 = 4.0;
  double a2 = -4.0, b2 = 4.0;
  double eps = 1.0e-4;
  double c = 0.9;
  double f = 0.9;
  DE de = new DE(func,a1,b1,a2,b2,N,eps,c,f);
  de.Calculate();
  de.Print();
  Console.ReadLine();
  }
  }
}
```

The results among runs vary due to the stochastic characteristics of the method. Typical output results:

```
Xbest = 0.228278198063108
Ybest = -1.62553510473607
Fbest = -6.55113333283091
Iter = 76
```

Bibliography

Steeb W.-H., Hardy Y., Hardy A. and Stoop R.
Problems and Solutions in Scientific Computing with C++ and Java Simulations
World Scientific Publishing, Singapore (2004)

Steeb W.-H.
The Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithm, Gene Expression Programming, Wavelets, Fuzzy Logic, fifth edition
World Scientific Publishing, Singapore 2011
ISBN 978-981-4335-77-5
`http://www.worldscibooks.com/chaos/8050.html`

Hardy Y., Kiat Shi Tan and Steeb W.-H.
Computer Algebra with SymbolicC++

World Scientific Publishing, Singapore 2008
ISBN-13: 978-981-283-360-0
`http://www.worldscibooks.com/mathematics/6966.html`

Steeb W.-H.,
Mathematical Tools in Signal Processing with C++ and Java Simulations
World Scientific Publishing, Singapore 2005
ISBN 981 256 500 0
`http://www.worldscibooks.com/engineering/5939.html`

Index