Problems and Solutions for Bit and String Manipulations

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

Preface

The purpose of this book is to supply a collection of problems in bitwise operations and string manipulations.

The material was tested in our lectures given around the world.

Any useful suggestions and comments are welcome.

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

e-mail addresses of the authors:

steebwilli@gmail.com

Home pages of the authors:

http://issc.uj.ac.za

Contents

Chapter 1

Basic Bitwise Operations

1.1 Introduction

Bit is short for binary digit with either of the two digits 0 and 1 in the binary number system. The bit is the smallest unit of storage in a binary system. Binary refers to base 2 arithmetic using the digits 0 and 1. Thus a bit is a binary digit (i.e. a digit in the binary number system). It is the most basic unit of information in digital computing.

A *boolean function* is a function f with domain $\{0,1\}^n$ and range $\{0,1\}$, for some positive integer n. Here $\{0,1\}^n$ denotes the n-fold Cartesian product of the set $\{0,1\}$ with itself, that is, the set of binary n-tuples. Thus we also write

$$f : \{0,1\}^n \to \{0,1\}.$$

There are 2^{2^n} boolean function of n variables.

We also have a set of m boolean functions $f_1,\ f_2,\ \ldots,\ f_m$, where $f_j : \{0,1\}^n \to \{0,1\}$ and $j = 1, 2, \ldots, m$. We also write

$$\mathbf{f} : \{0,1\}^n \to \{0,1\}^m.$$

There are 2^{m2^n} functions $\mathbf{f} : \{0,1\}^n \to \{0,1\}^m$.

A *truth table* is a tabular description of a combinational circuit (such as an AND-gate, OR-gate, NAND-gate) listing all possible states of the input

variables together with a statement of the output variable(s) for each of those possible states. The truth table for the AND-gate, OR-gate, XOR-gate and NOT-gate are

AND		
0	0	0
0	1	0
1	0	0
1	1	1

OR		
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
0	1
1	0

The NAND-gate is an AND-gate followed by a NOT-gate. The NOR-gate is an OR-gate followed by a NOT-gate. Both are *universal gates*, i.e. all other gates can be built from these gates.

Notation. In the text we use for the AND-operation $\cdot$, we use for the OR-operation $+$, we use for the XOR-operation $\oplus$ and we use for the NOT-operation $\overline{x}$.

Programming Languages. In C++, C, Java, C# and Perl the bitwise AND is the ampersand operator `&`. The bitwise OR is `|`. The bitwise XOR is `^`. The bitwise NOT (one complement) is `~`. In C and C++ the order of *precedence* of the bitwise operators is `&`, `^`, `|`.

The shift operations allow bits to be moved to the left or right in a word. There are three types of shift operations: logical, rotate and arithmetic. The logical shift moves bits to the left or right. The bits which fall off the end of the word are discarded and the word is filled with 0's from the opposite end. For example a logical right shift of the 8 bit binary number `1000 1011` provides `0100 0101`. Shift instructions include a repeat value, which is the number of times the single bit shift operation is repeated. A rotate operation (not implemented in C++, C, Java, C# and Perl) is a circular shift in which no bits are discarded. An arithmetic right shift is similar to a logical right shift except that the leftmost bits are filled with the sign bit of the original number instead of 0's. For example, an arithmetic right shift of the 8 bit number `1000 1011` provides `1100 0101`. `>>` is the right shift operation in C++, C, Java, C# and Perl. `<<` is the left shift operation in these languages. In C++ and C they can be applied to integral operands, that is `char`, `short`, `int`, and `long`, whether signed or unsigned. The size of `char` is 8 bits, the size of `short` is 16 bits, and the size of `int` and `long` is 32 bits. The sign bit is at the left most position, for example for `int` it is at position 31 (counting from 0).

Hex notation is indicated in C, C++, Java and Perl as `0x`. For example `0xFF` is the number 255 (base 10).

The AND-operation performs a bit-by-bit logical AND of two bitstrings of the same length.
The OR-operation performs a bit-by-bit logical OR of two bitstrings of the same length.
The XOR-operation performs a bit-by-bit logical exclusive OR of two bitstrings of the same length.
The MOD-operation returns the remainder (modulus) from dividing two integer numbers.
The NOT-operation performs a bit-by-bit complement of a bitstring.
SHL shifts the value of the bitstring to the left count bits. A negative count causes the data to be shifted the opposite way.
SHR shifts the value of the bitstring to the right count bits. A negative count causes the data to be shifted the opposite way.

1.2 Quickies

Problem 1. Let $x \in \{0,1\}$.
(i) Find $x \cdot x$, $x \oplus x$, $x + x$.
(ii) Find $x \cdot \bar{x}$, $x \oplus \bar{x}$, $x + \bar{x}$.
(iii) Find $x \cdot 0$, $x \cdot 1$, $x \oplus 0$, $x \oplus 1$, $x + 0$, $x + 1$.

Problem 2. Let $x, y \in \{0,1\}$.
(i) Solve the boolean equation

$$x \cdot y = x + y.$$

(ii) Solve the boolean equation

$$x \oplus y = x \cdot y.$$

(iii) Solve the boolean equation

$$x \oplus y = x + y.$$

Problem 3. Let $x_1, x_2, x_3, x_4 \in \{0,1\}$.
(i) Is

$$(x_1 + x_2) \cdot (x_3 + x_4) = x_1 \cdot x_3 + x_1 \cdot x_4 + x_2 \cdot x_3 + x_2 \cdot x_4 \,?$$

(ii) Is

$$(x_1 + x_2) \oplus (x_3 + x_4) = x_1 \oplus x_3 + x_1 \oplus x_4 + x_2 \oplus x_3 + x_2 \oplus x_4 \,?$$

Problem 4. Let $x_1, x_2, x_3 \in \{\,0,1\,\}$. Let $\oplus$ be the XOR-operation and $\cdot$ be the AND-operation. Is

$$(x_1 \cdot x_2) \oplus x_3 = x_1 \cdot (x_2 \oplus x_3) \,?$$

Problem 5. Let $a, b, c, d \in \{\,0,1\,\}$. Consider the boolean function $f : \{0,1\}^4 \to \{0,1\}$

$$f(a,b,c,d) = a + b \oplus c \cdot d.$$

In what order of precedence are the operations AND $\cdot$, XOR $\oplus$, and OR $+$ applied? Find $f(1,1,1,1)$.

Problem 6. (i) Can the boolean function $f : \{0,1\}^2 \to \{0,1\}$

$$f(x,y) = x + (x \cdot y)$$

be simplified? Here + denotes the XOR-operation and $\cdot$ the AND-operation.
(ii) Can be boolean function $f : \{0,1\}^2 \to \{0,1\}$

$$f(x,y) = x \cdot (x+y)$$

be simplified? Here + denotes the XOR-operation and $\cdot$ the AND-operation.

Problem 7. Let $x, y, z \in \{0,1\}$. Is

$$(x \cdot y) + (y \cdot z) + (z \cdot x) = (x \cdot y) \oplus (y \cdot z) \oplus (z \cdot x)$$

where $\cdot$ denotes the AND-operation, + the OR-operation and $\oplus$ the XOR-operation?

Problem 8. Let $a, b, c \in \{\,0,1\,\}$. Is

$$(a \oplus b) \cdot c = a \oplus (b \cdot c)\,?$$

Problem 9. Let $a, b, x, y \in \{0,1\}$. Solve the bitwise equation

$$a \oplus b = x \cdot y.$$

Problem 10. (i) Let x, y, z be bitstrings of the same length. Is the XOR-operation *associative*, i.e.

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)\,?$$

(ii) Let x and y be arbitrary bitstrings of the same length. Let $\oplus$ be the XOR operation. Calculate

$$(x \oplus y) \oplus y.$$

Problem 11. Let $x, y, z \in \{0,1\}$. Is

$$(x \oplus y) \cdot z = (x \cdot z) \oplus (y \cdot z)\,?$$

Problem 12. Let $\odot$ be the exclusive-NOR operation, i.e. $x \odot y = 1$ if and only if $x = y$ for the boolean variables x and y. Find the solutions of

$$x + y = x \odot y, \quad x \cdot y = x \odot y, \quad x \oplus y = x \odot y.$$

Problem 13. Show that the bitwise expression

$$b \oplus (a \cdot c)$$

contains the AND-gate, XOR-gate, NOT-gate and FANOUT. First write down the truth table.

Problem 14. Given the truth table

row	x_2	x_1	x_0	z
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Find the boolean expression (*sum of products*) for this truth table. Can the expression be simplified?

Problem 15. Given k-bit inputs and m-bit outputs. How many boolean function are there?

Problem 16. The bitwise NOT-operation is defined by `0 -> 1` and `1 -> 0`. Thus in a bitstring the NOT-operation replaces 0's by 1's and 1's by 0's. How can the bitwise NOT-operation be implemented using the XOR-operation?

Problem 17. Can the expression

$$E = \overline{x} \cdot \overline{z} + x \cdot y + \overline{x} \cdot \overline{y} + y \cdot \overline{z}$$

be simplified?

Problem 18. The truth table of a typical *encoder* with inputs a, b, c, d and outputs $c0$, $c1$ is

a	b	c	d	$c0$	$c1$
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Find the Boolean expression for this truth table.

Problem 19. The truth table of a *full adder* with inputs a, b, *cin* and outputs y, *cout* is

a	b	cin	y	$cout$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Find the Boolean expression for this truth table.

Problem 20. A 1-bit *full adder* is a circuit with three 1-bit inputs (the bits to be added) and two 1-bit outputs (the sum and the carry). It is given by

$$\begin{aligned}\text{sum} &= (a \oplus b) \oplus c \\ \text{carry} &= a \cdot b + a \cdot c + b \cdot c = a \cdot b + (a + b) \cdot c = a \cdot b + (a \oplus b) \cdot c.\end{aligned}$$

Find the thruth table.

Problem 21. We define the *binary dot product* of two bitstrings $\mathbf{x}$ and $\mathbf{y}$ of the same length n as

$$\mathbf{x} \cdot \mathbf{y} := (x_1 y_1 + x_2 y_2 + \cdots + x_n y_n) \bmod 2.$$

Let $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ be bitstrings of the same length. Verify that

$$(\mathbf{x} \oplus \mathbf{y}) \cdot \mathbf{z} \equiv (\mathbf{x} \cdot \mathbf{z}) \oplus (\mathbf{y} \cdot \mathbf{z}).$$

Problem 22. Let $a, b, c \in \{0, 1\}$. Find all solutions of

$$a \oplus b \oplus c = a \cdot b \cdot c.$$

Problem 23. Prove the following statements by (i) using a truth table, (ii) using the properties of boolean algebra (algebraic proof).

$$\text{(a)} \quad 1 + a \equiv 1.$$

$$\text{(b)} \quad \overline{a} \oplus \overline{b} \equiv a \oplus b$$

$$\text{(c)} \quad (a \cdot \overline{b}) \oplus (a \cdot b) \equiv a \cdot (\overline{b} \oplus b).$$

Problem 24. Let $x \in \{\, 0, 1 \,\}^{2n}$ and

$$Y(x) := \{\, y \in \{0, 1\}^{2n} \,:\, h(x, y) = n \,\}$$

$$C(x,y) := \{\, (x,y) \,:\, x \in \{\,0,1\}^{2n},\, y \in Y(x)\,\}$$

where $h(x,y)$ denotes the *Hamming distance* between x and y.

(a) Determine $|Y(x)|$, where $||$ denotes the cardinality, i.e. the number of elements.
(b) Determine $|C(x,y)|$, where $||$ denotes the cardinality, i.e. the number of elements.
(c) Let (x,y). How many unique pairs are (x,y) in $C(x,y)$?
(d) Which group properties hold for $Y(x_0)$ where $x_0 = (0,0,\ldots,0)$?

Problem 25. The NAND-gate is a universal gate. The XOR-gate can be represented with four NAND-gates. Use multi expression programming to implement the XOR-gate using NAND-gates.

Problem 26. Show that the XOR-gate can be built from 4 NAND-gates. Show that the AND-gate can be built from 2 NAND-gates. Show that the OR-gate can be built from 3 NAND-gates. Show that the NOR-gate can be built from 4 NAND-gates.

Problem 27. Consider the one-dimensional 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 $N = 6$ bits and $x = 1/8$. Calculate the orbit $f(x)$, $f(f(x))$, $f(f(f(x)))$, ... with this precision. Discuss.

Problem 28. Find the truth table for the boolean function

$$f(a,a',b,b') = (a \cdot b') \oplus (a' \cdot b).$$

Problem 29. Consider the boolean function

$$f(x_1,x_2,x_3) = x_1 \oplus x_2 \oplus x_3.$$

Find the *disjunctive normal form.*

Problem 30. Show that every boolean function $f : \{0,1\}^n \to \{0,1\}$ can be expanded as follows

$$f(x_1, x_2, \ldots, x_n) = x_1 \cdot f(1, x_2, \ldots, x_n) + \overline{x}_1 \cdot f(0, x_2, \ldots, x_n).$$

Problem 31. Apply the expansion theorem given at the previous exercise repeatedly to each variable of

$$f(x_1, x_2, x_3) = x_1 \cdot \overline{x}_2 + x_2 \cdot x_3$$

to obtain its disjunctive normal form.

Problem 32. Consider the first-order discrete time dynamical system

$$x_{k+1} = 2x_k \mod 1 \quad k = 0, 1, 2, \ldots$$

and

$$s_k = \begin{cases} 1 \text{ if } x_k \geq 0.5 \\ 0 \text{ if } x_k < 0.5 \end{cases}$$

where $x_0 \in [0, 1]$. We call $\mathbf{s} = s_0 s_1 s_2 \ldots$ the output symbol. Show that if $x_0 \in [0.78125, 0.8125]$ then the output coincide for the first three bits.

Problem 33. Let n be the number of discrete symbols s_1, s_2, ..., s_n that can be used. Let m be the length of the message string. Find the number M of messages. Then consider the special case $n = m = 2$.

Problem 34. 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 $m = 4$, $m_1 = m_2 = 2$.

Problem 35. The D-type latch works as follows. When the control signal latch-enable (LEN) is high, the latch is in the transparent mode and the input signal $\bar{D}$ is available at the output. When the LEN signal is low, the input data is latched to the output and is retained until LEN goes back to high.
(i) Give the truth table from this description.
(ii) Give the boolean equation for this latch.
(iii) Give the circuit.

Problem 36. Consider two `unsigned int` (32 bits) n and $n + 1$. What is the condition that the parity of n and $n + 1$ are the same? For example 1 and 2 have the same parity but 2 and 3 not.

Problem 37. Given the boolean function

$$f(a,b,c,d) = a \cdot d + (a \cdot c + b) \cdot (c \cdot d + e).$$

Find the truth table. Discuss. Write a C++ program that generates the truth table.

Problem 38. Write the boolean function

$$f(x,y,z) = \overline{(x \cdot \bar{y}) + x \cdot z)} + \bar{x}$$

in disjunctive normal form.

Problem 39. Consider the boolean function $f : \{0,1\}^3 \to \{0,1\}$

$$f(a,b,c) = a \cdot b \cdot \bar{c} + \bar{a} \cdot c + \bar{a} \cdot \bar{b}.$$

Find the truth table.

Problem 40. Simplify the boolean expression

$$f(x,y,z) = x \cdot (\bar{x} + y) + y \cdot (y + z) + y.$$

Problem 41. Given the boolean function

$$f(x_1,x_2,x_3,x_4) = (x_1 + x_2) \cdot (x_3 + x_4).$$

Give the truth table.

Problem 42. Consider the truth table

I_1	I_2	I_3	O
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Find a boolean expression.

Problem 43. Let $a, b \in \{0,1\}$. Find all solutions of the equation $a \oplus b = a \cdot b$, where $\oplus$ denotes the XOR-operation and $\cdot$ the AND-operation.

Problem 44. Let $a, b, c, d \in \{0, 1\}$. Find all solutions of the equation $a \cdot b = c \oplus d$, where $\oplus$ denotes the XOR-operation and $\cdot$ the AND-operation.

Problem 45. Let i_1, i_2, i_3 be the input bits and o_1, o_2, o_3 be the output bits. We say that the input bits and the output bits pass the GHZ test if they satisfy the system of boolean equation

$$\begin{aligned} i_1 \oplus i_2 \oplus i_3 &= 0 \\ o_1 \oplus o_2 \oplus o_3 \oplus (i_1 \vee i_2 \vee i_3) &= 1. \end{aligned}$$

Find all solutions of this system of boolean equation. Here $\oplus$ denotes the XOR-operation and $\vee$ denotes the OR-operation.

Problem 46. After the ASCII table the capital A is identified with the integer 65 (base 10) and the small a is identified with the integer 97 (base 10). Write down these two numbers in binary (8 bits) and apply the XOR-operation. Discuss.

Problem 47. Prove (i) using a truth table, (ii) using the properties of boolean algebra (algebraic proof)
(a) $1 + x = 1$
(b) $\bar{x} \oplus \bar{y} = x \oplus y$
(c) $(x \cdot \bar{y}) \oplus x = x \cdot y$
(d) that $\oplus$ is associative.

Problem 48. Let I, J, K, M be `unsigend int`. Write a C++ program that implements

```
((I XOR J) AND (K XOR M)) OR ((I XOR K) AND (J XOR M)).
```

Discuss.

Problem 49. Let $a, b, c, x, y, z \in \{0, 1\}$. Solve the boolean equation

$$a \cdot (b \cdot c) = x + (y + z).$$

1.3 Explain the Output of the C++ Program

1.3.1 Bitwise Operations

Problem 50. In the following C++ program that bitwise AND `&` is applied. What is the output?

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

int main(void)
{
  int x = 17;
  int r = x & (-x);
  cout << "r = " << r << endl;
  x = 101;
  r = x & (-x);
  cout  << "r = " << r << endl;
  x = -5;
  r = x & (-x);
  cout << "r = " << r << endl;
  return 0;
}
```

Problem 51. What is the output of the following C++ program?

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

int main(void)
{
  unsigned int i = 10; unsigned int j = 11;
  unsigned int k = 12; unsigned int l = 13;
  unsigned r = ((i^j) & (k*l)) | ((i^k) & (j^l));
  cout << "r = " << r << endl;
  return 0;
}
```

Problem 52. In the following C++ program the bitwise XOR `^` is used. What is the output?

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

```
int main(void)
{
  unsigned int a = 10; unsigned int b = 9;
  unsigned int x = 10;
  x = a^b^x;
  cout << "x = " << x << endl;
  a = 7; b = 9; x = 9;
  x = a^b^x;
  cout << "x = " << x << endl;
  return 0;
}
```

Problem 53. In the following C++ program we apply the NOT operation ~ and the OR operation |. What is the output?

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

int main(void)
{
 unsigned int x = 8345; unsigned int y = 34512;
 unsigned int r1 = x & y;
 unsigned int r2 = ~(~x | ~y);
 if(r1==r2) cout << "true"; else cout << "false";
 return 0;
}
```

Problem 54. What is output of the following C++ program using the AND and NOT operations

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

int main(void)
{
 unsigned int i = 476; unsigned int j = 677;
 unsigned int r = i & ~j;
 cout << "r = " << r << endl;
 return 0;
}
```

Problem 55. What is the output of the following C++ program?

```
// output.cpp
```

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

int main(void)
{
  unsigned int i=1; unsigned int j=2;
  unsigned int k=3;
  unsigned int f1 = (((i) & (j)) | ((~i) & (k)));
  unsigned int f2 = (((i) & (k)) | ((j) & (~k)));
  unsigned int f3 = ((i) ^ (j) ^ (k));
  unsigned int f4 = ((j) ^ ((i) | (~k)));
  cout << "f1 = " << f1 << endl;
  cout << "f2 = " << f2 << endl;
  cout << "f3 = " << f3 << endl;
  cout << "f4 = " << f4 << endl;
  return 0;
}
```

Problem 56. The following C++ program uses the AND operation. What is the output?

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

int main(void)
{
   int x = 22; int y = 18;
   int t = ((x-y) & -(x < y));
   int r1 = y + t;
   cout << "r1 = " << r1 << endl;
   int r2 = x-t;
   cout << "r2 = " << r2 << endl;
   return 0;
}
```

Problem 57. The following C++ uses the AND operation. Note that `!` is the logical NOT. What is the output?

```
// parity1.cpp
#include <iostream>
using namespace std;

int main(void)
{
 unsigned int v = 19;
 bool p = false;
```

```
 while(v != 0)  // short-cut while(v)
 { p = !p; v = v & (v-1); }
 cout << "p = " << p << endl;
 return 0;
}
```

Problem 58. The following C++ code uses the AND-operation. Here `!` is the logical NOT. What is the output?

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

int main(void)
{
  unsigned int i1 = 2345678;
  bool b1 = !(i1 & (i1-1)) && (i1 > 0);
  cout << "b1 = " << b1 << endl;
  unsigned int i2 = 65536;
  bool b2 = !(i2 & (i2-1)) && (i2 > 0);
  cout << "b2 = " << b2 << endl;
  unsigned int i3 = 0;
  bool b3 = !(i3 & (i3-1)) && (i3 > 0);
  cout << "b3 = " << b3 << endl;
  return 0;
}
```

Problem 59. The following C++ code use the NOT and AND-operation. What is the output?

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

int main(void)
{
 unsigned int j = 4236571;
 unsigned int r = j & ~0xFF;
 cout << "r = " << r << endl;
 return 0;
}
```

Problem 60. In the following C++ program we use the XOR and NOT-operation. What is the output?

```
// XORNOT.cpp
```

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

int main(void)
{
 unsigned int j;
 unsigned int r = j ^ (~j);
 cout << "r = " << r << endl;
 return 0;
}
```

Problem 61. The following program uses the OR operation. What is the output?

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

int main(void)
{
 unsigned int v = 19;
 unsigned int r = v | (v + 1);
 cout << "r = " << r << endl;
 return 0;
}
```

1.3.2 Shift Operations

Problem 62. The following C++ program uses the shift operation and the XOR operation. What is the output?

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

int main(void)
{
 unsigned int x = 7;
 unsigned int y;
 y = x^(x >> 1);
 y = y^(y >> 2);
 y = y^(y >> 4);
 y = y^(y >> 8);
 y = y^(y >> 16);
 cout << "y = " << y << endl;
 unsigned int r = y & 1;
 cout << "r = " << r << endl;
```

```
 return 0;
}
```

Problem 63. What is the output of the following C++ program?

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

unsigned int p(unsigned int v)
{
  v ^= v >> 16;
  v ^= v >> 8;
  v ^= v >> 4;
  v &= 0xf;
  return (0x6996 >> v) & 1;
}

int main(void)
{
  unsigned int v1 = 8;
  unsigned int r1 = p(v1);
  cout << "r1 = " << r1 << endl;
  unsigned int v2 = 9;
  unsigned int r2 = p(v2);
  cout << "r2 = " << r2 << endl;
  unsigned int v3 = 101;
  unsigned int r3 = p(v3);
  cout << "r3 = " << r3 << endl;
  return 0;
}
```

Problem 64. The following C++ program uses the shift-operation. What is the output?

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

unsigned int leading(unsigned int x)
{
  if(x==0) return 32;
  unsigned int n = 0;
  if(x <= 0x0000FFFF) { n += 16; x = x << 16; }
  if(x <= 0x00FFFFFF) { n += 8; x = x << 8; }
  if(x <= 0x0FFFFFFF) { n += 4; x = x << 4; }
  if(x <= 0x3FFFFFFF) { n += 2; x = x << 2; }
```

```
  if(x <= 0x7FFFFFFF) { n += 1; }
  return n;
}

int main(void)
{
  unsigned int x = 4;
  unsigned int r = leading(x);
  cout << "r = " << r << endl;
  x = 100;
  r = leading(x);
  cout << "r = " << r << endl;
  x = 255;
  r = leading(x);
  cout << "r = " << r << endl;
  return 0;
}
```

Problem 65. The following C++ program utilizes the AND, OR and shift operation. What is the output?

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

unsigned int reversing(unsigned int x)
{
 x = (x & 0x55555555) << 1 | (x & 0xAAAAAAAA) >> 1;
 x = (x & 0x33333333) << 2 | (x & 0xCCCCCCCC) >> 2;
 x = (x & 0x0F0F0F0F) << 4 | (x & 0xF0F0F0F0) >> 4;
 x = (x & 0x00FF00FF) << 8 | (x & 0xFF00FF00) >> 8;
 x = (x & 0x0000FFFF) << 16 | (x & 0xFFFF0000) >> 16;
 return x;
}

int main(void)
{
 unsigned int x1 = 0;
 unsigned int r1 = reversing(x1);
 cout << "r1 = " << r1 << endl;
 unsigned int x2 = 1;
 unsigned int r2 = reversing(x2);
 cout << "r2 = " << r2 << endl;
 return 0;
}
```

Problem 66. The following C++ program uses the shift and OR opera-

tion. What is the output of the following C++ program?

```
// isqrt.cpp

#include <iostream>
using namespace std;

unsigned int isqrt(unsigned int x)
{
  unsigned int m, y, b;
  m = 0x40000000;
  y = 0;
  while(m != 0)
  {
  b = y | m;
  y = y >> 1;
  if(x >= b) { x = x-b; y = y | m; }
  m = m >> 2;
  }
  return y;
}

int main(void)
{
  unsigned int x1 = 99;
  unsigned int r1 = isqrt(x1);
  cout << "r1 = " << r1 << endl;
  unsigned int x2 = 100;
  unsigned int r2 = isqrt(x2);
  cout << "r2 = " << r2 << endl;
  return 0;
}
```

Problem 67. The following program uses the shift operation. What is the output?

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

unsigned int cr(unsigned int x)
{
 int s = 30;
 unsigned int y, b;
 y = 0;
 while(s >= 0)
 {
 y = y << 1;
```

```
 b = (3*y*(y+1) + 1) << s;
 s -= 3;
 if(x >= b) { x -= b; y += 1; } // end if
 } // end while
 return y;
}

int main(void)
{
 unsigned int x = 100;
 unsigned int r1 = cr(x);
 cout << "r1 = " << r1 << endl; // =>
 x = 200;
 unsigned int r2 = cr(x);
 cout << "r2 = " << r2 << endl; // =>
 return 0;
}
```

Problem 68. What is the output of the following C++ program?

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

unsigned int shiftOR(unsigned int v)
{
  v = v | (v >> 1);
  v = v | (v >> 2);
  v = v | (v >> 4);
  v = v | (v >> 8);
  v = v | (v >> 16);
  return v - (v >> 1);
}

int main(void)
{
 unsigned int j = 66;
 unsigned int result = shiftOR(j);
 cout << "result = " << result << endl;
 return 0;
}
```

Problem 69. The function `int f(int)` in the following C++ program uses the XOR operation and shift operation. What is the output?

```
// absolute.cpp
#include <iostream>
```

```
using namespace std;

int f(int i)
{
  int t = sizeof(int);
  int r;
  r = (i^(i >> 31))-(i >> 31);
  return r;
}

int main(void)
{
  int n1 = -87;
  int r1 = f(n1);
  cout << "r1 = " << r1 << endl;
  int n2 = 99;
  int r2 = f(n2);
  cout << "r2 = " << r2 << endl;
  return 0;
}
```

Problem 70. The following C++ program uses the AND operation and shift operation. What is the output?

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

const unsigned char Table[] =
{ 0,1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
  1, 2, 2, 3, 2, 3, 3, 4, 2, 3, 3, 4, 3, 4, 4, 5,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
  2, 3, 3, 4, 3, 4, 4, 5, 3, 4, 4, 5, 4, 5, 5, 6,
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
  3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7,
  4, 5, 5, 6, 5, 6, 6, 7, 5, 6, 6, 7, 6, 7, 7, 8 };

int main(void)
```

```
{
  unsigned int v1 = 14;
  unsigned int c1 = Table[v1 & 0xff] + Table[(v1 >> 8) & 0xff]
                  + Table[(v1 >> 16) & 0xff] + Table[v1 >> 24];
  cout << "c1 = " << c1 << endl;
  unsigned int v2 = 234;
  unsigned int c2 = Table[v2 & 0xff] + Table[(v2 >> 8) & 0xff]
                  + Table[(v2 >> 16) & 0xff] + Table[v2 >> 24];
  cout << "c2 = " << c2 << endl;
  return 0;
}
```

Problem 71. What is the output of the following C++ program

```
// output1.cpp
#include <iostream>
using namespace std;

unsigned int add(unsigned int x,unsigned int y)
{
  unsigned int low = (x & 0xffff) + (y & 0xfff);
  unsigned int high = (x >> 16) + (y >> 16) + (low >> 16);
  return (high << 16) | (low & 0xffff);
}

int main(void)
{
  unsigned int x = 17; unsigned int y = 23;
  unsigned int result = add(x,y);
  cout << "result = " << result << endl;
  return 0;
}
```

Problem 72. The following C++ program uses the XOR, AND and shift operation. What is the output?

```
// parity2.cpp
#include <iostream>
using namespace std;

unsigned int parity(unsigned char v)
{
  v ^= v >> 4;
  v &= 0xf;      // short cut for v = v & 0xf
  return ((0x6996 >> int(v)) & 1);
}
```

```
int main(void)
{
 unsigned char v1 = 19;
 unsigned int r1 = parity(v1);
 cout << "r1 = " << r1 << endl;
 unsigned char v2 = 18;
 unsigned int r2 = parity(v2);
 cout << "r2 = " << r2 << endl;
 return 0;
}
```

Problem 73. The following C++ program uses the shift and AND operation. What is the output?

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

int main(void)
{
  unsigned int i = 234;
  unsigned int c;
  for(c=0;i>0;i>>=1) { c += i & 1; }
  cout << "c = " << c << endl;
  return 0;
}
```

Problem 74. The following C++ operation applies the shift operation. What is the output?

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

int main(void)
{
  int i1 = 345271;
  int r1 = (i1 >> 31);
  cout << "r1 = " << r1 << endl;
  int i2 = -471273;
  int r2 = (i2 >> 31);
  cout << "r2 = " << r2 << endl;
  return 0;
}
```

Problem 75. The following C++ program uses the shift operation. What is the output?

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

int main(void)
{
  unsigned int i1 = 12;
  unsigned int r1 = (i1 << 3) - i1;
  cout << "r1 = " << r1 << endl;
  unsigned int i2 = 12;
  unsigned int r2 = (i2 << 4) + i2;
  cout << "r2 = " << r2 << endl;
  unsigned int i3 = 12;
  unsigned int r3 = (i3 << 6) + i3;
  cout << "r3 = " << r3 << endl;
  unsigned int i4 = 12;
  unsigned int r4 = (i4 << 3) + (i4 << 2) + i4;
  cout << "r4 = " << r4 << endl;
  return 0;
}
```

Problem 76. The following C++ program uses the shift operation and AND-operation. What is the output?

```
// parity3.cpp
#include <iostream>
using namespace std;

unsigned int parity(unsigned int v)
{
  v ^= v >> 16;
  v ^= v >> 8;
  v ^= v >> 4;
  v &= 0xf;
  return ((0x6996 >> v) & 1);
}

int main(void)
{
  unsigned int v1 = 10;
  unsigned int r1 = parity(v1);
  cout << "r1 = " << r1 << endl;
  unsigned int v2 = 11;
  unsigned int r2 = parity(v2);
  cout << "r2 = " << r2 << endl;
  return 0;
}
```

Problem 77. What is the output of the following C++ program

```
// Hammingunsigned1.cpp
#include <iostream>
using namespace std;

int main(void)
{
   unsigned int x = 4;
   unsigned int y = 4294967295;
   unsigned int r = x^y;
   int j = 31;
   int count = 0;
   while(j >= 0)
   {
   if((r & 1)==1) count++;
   r = r >> 1;
   j--;
   }
   cout << "count = " << count;
   return 0;
}
```

Problem 78. The following C++ program uses the XOR, AND, OR and shift operations. What is the output?

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

unsigned int swap(unsigned int v,unsigned int i,unsigned int j,
                  unsigned int n)
{
 unsigned int temp = ((v >> i)^(v >> j)) & ((1 << n)-1);
 return (v^((temp << i)|(temp << j)));
}

int main(void)
{
 unsigned int i = 2, j = 5;
 unsigned int n = 2;
 unsigned int v = 24;
 unsigned int result = swap(v,i,j,n);
 cout << "result = " << result << endl;
 return 0;
}
```

Problem 79. What is the output of the following C++ program?

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

int main()
{
  unsigned int i = 4294967295;
  unsigned int j = 1;
  unsigned int r = i+j;
  cout << "r = " << r << endl;
  return 0;
}
```

Use the binary representation of the number 4294967295 and the number 1.

Problem 80. What is the output of the following C++ program?

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

int main(void)
{
 unsigned int i = 44;
 unsigned int i0 = i & 0xFFFF;
 unsigned int i1 = i >> 0x10;
 unsigned int r0 = i0*0xAAAB;
 unsigned int t = i1*0xAAAB + (r0 >> 0x10);
 unsigned int r1 = t & 0xFFFF;
 unsigned int r2 = t >> 0x10;
 r1 += i0*0xAAAA;
 unsigned p = i1*0xAAAA + r2 + (r1 >> 0x10);
 unsigned result = p >> 1;
 cout << "result = " << result;
 return 0;
}
```

Problem 81. In C and C++ the precedence of the bitwise operators is as follows:

```
0  ~  one's complement
1  &  bitwise AND
2  ^  bitwise XOR
3  |  bitwise OR
```

What is the output of the following C++ program?

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

int main(void)
{
 unsigned int a = 10;
 unsigned int b = 11;
 unsigned int c = 12;
 unsigned int r1 = a & b | c;
 cout << "r1 = " << r1 << endl;
 unsigned int r2 = a & (b | c);
 cout << "r2 = " << r2 << endl;
 return 0;
}
```

1.4 bitset class

Problem 82. The following program uses the `bitset` class of C++ and utilizes pointers. What is the output of the program?

```
// bitset1.cpp
#include <iostream>
#include <bitset>
using namespace std;

const unsigned int n = 32;

void swap(bitset<n>* b1,bitset<n>* b2)
{ *b1 = (*b1)^(*b2); *b2 = (*b2)^(*b1); *b1 = (*b1)^(*b2); }

int main()
{
 bitset<n> bs1;
 bs1.flip(4);
 cout << "bs1 = " << bs1 << endl;
 bitset<n>* pbs1 = new bitset<n>;
 pbs1 = &bs1;
 cout << "pbs1 = " << pbs1 << endl;
 bitset<n> bs2;
 bs2.flip(7);
 cout << "bs2 = " << bs2 << endl;
 bitset<n>* pbs2 = new bitset<n>;
 pbs2 = &bs2;
 swap(pbs1,pbs2);
 cout << "after swapping:" << endl;
 cout << "bs1 = " << bs1 << endl;
 cout << "bs2 = " << bs2 << endl;
 return 0;
}
```

Problem 83. The following program uses the `bitset` class of C++. A pointer to `bitset` is declared. What is the output of the following program?

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

int main(void)
{
 const unsigned int n = 32;
 bitset<n>* bp = new bitset<n>;
 (*bp).set();
```

```
 (*bp).flip(4);
 cout << *bp << endl;
 delete bp;
 return 0;
}
```

Problem 84. The following program uses the `bitset` class of C++. What is the output of the program?

```
// uinttobitset1.cpp
#include <iostream>
#include <bitset>
using namespace std;

void convert(bitset<32>& s,unsigned int i,int n)
{
 int count = 0;
 while(count < n)
 {
 int t = 1 & i;
 if(t != 0) s.set(count,1);
 else s.set(count,0);
 i = i >> 1;
 count++;
 }
}  // end convert

int main(void)
{
 const unsigned int n = 32;
 bitset<n> s;
 unsigned int i = 133;
 convert(s,i,n);
 cout << "s = " << s << endl;
 return 0;
}
```

Problem 85. (i) Given an unsigned int number (32 bits). Write a C++ program that converts it to a bitstring of the `bitset` class.
(ii) Given an signed int number (32 bits). Write a C++ program that converts it to a bitstring of the `bitset` class.
(iii) Given a floating point number `float` (32 bits). Write a C++ program that converts it to a bitstring of the `bitset` class.
(iv) Given a floating point number `double` (64 bits). Write a C++ program that converts it to a bitstring of the `bitset` class.

Problem 86. (i) Given a bitstring of the `bitset` class of length 32. Write a C++ program that converts the bitstring into an unsigned int. First one has to check whether the bitstring refers to NaN.
(ii) Given a bitstring of the `bitset` class of length 32. Write a C++ program that converts the bitstring into a signed int. First one has to check whether the bitstring refers to NaN.
(iii) Given a bitstring of the `bitset` class of length 32. Write a C++ program that converts the bitstring into a float. First one has to check whether the bitstring refers to NaN.
(iv) Given a bitstring of the `bitset` class of length 64. Write a C++ program that converts the bitstring into a `double`. First one has to check whether the bitstring refers to NaN.

Problem 87. Show that

$$67_{10} = 74_9 = 103_8 = 124_7 = 151_6 = 232_5 = 1003_4 = 2111_3 = 1000011_2$$

and $67_{10} = 43_{16}$.

Chapter 2

Advanced Bitwise Manipulations

2.1 Write a C++ Program

Problem 1. Let $a, b, c, x, y, z \in \{0, 1\}$. Let $+$ be the bitwise OR and $\cdot$ be the bitwise AND. Write a C++ program that finds all solutions of the equation

$$a + b + c = x \cdot y \cdot z.$$

Count the number of solutions.

Problem 2. Let $a, b, c, x, y, z \in \{0, 1\}$. Let $\oplus$ be the bitwise XOR and $\cdot$ be the bitwise AND. Write a C++ program that finds all solutions of the equation

$$a \oplus b \oplus c = x \cdot y \cdot z.$$

Count the number of solutions.

Problem 3. Let `x` and `y` be `unsigned int`'s. Write a function

```
unsigned int setbits(x,p,n,y)
```

that returns `x` with the `n` bits that begin at position `p` set to the rightmost `n` bits of `y`, leaving the other bits unchanged.

Problem 4. Given two `unsigned int x` and `y`. Write a C++ program that finds the *Hamming distance* between `x` and `y`. Apply the XOR operation.

Problem 5. Given a bitstring with an even number n of bits. Half $(n/2)$ of the bits are 0's and therefore the other half are 1's.
(i) In how many ways can we form such a bitstring for a given n.
(ii) Write a C++ program using `bitset<N>` that finds all the possible bitstrings for a given n and stores them in lexicographical order. For example, for $n = 2$ we have `01`, `10`. For $n = 4$ we have

```
0011   0101   0110   1001   1010   1100
```

Problem 6. Let x, y be bitstrings of the same length n. We define a *scalar product*

$$x \star y := (x_0 \cdot y_0) \oplus (x_1 \cdot y_1) \oplus \cdots \oplus (x_{n-1} \cdot y_{n-1}).$$

Write a C++ program that implements this scalar product.

Problem 7. Given two `unsigned int m` and `n`. Write a C++ function

```
bool check(unsigned int m,unsigned int n)
```

which returns `true` if `m` and `n` are both even and `false` otherwise. Apply bitwise operations.

Problem 8. Given an `unsigned int` (32 bits) in C++. Write a C++ program that reverses the bits.

Problem 9. Let `x` be an `unsigned int`. Write a function `invert(x,p,n)` that returns `x` with the `n` bits that begin at position `p` inverted (i.e. 1 is changed into 0 and vice versa), leaving others unchanged.

Problem 10. Let `x` be an `unsigned int`. Write a C++ function

```
unsigned int rightrot(x,n)
```

that returns the value of the integer `x` rotated to the right by `n` bit positions. Apply the shift operation.

Problem 11. Rewrite the following C++ program without the `if ... else` using bitwise operations. What is the program doing?

```
// iftoxor0.cpp

#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main(void)
{
  unsigned int a, b, x;
  srand((unsigned int) time(NULL));
  x = rand()%2;
  cout << "x = " << x << endl;
  a = 0; b = 1;
  if(x==a) x = b; else x = a;
  cout << "x = " << x << endl;
  return 0;
}
```

Problem 12. Let `i`, `j` be integers. The *doz function* is "difference or zero" and is defined as

$$\mathrm{doz}(i,j) := \begin{cases} i-j \text{ if } i \geq j \\ 0 \quad \text{if } i < j \end{cases}$$

Write a C++ program using bitwise operations that implements this function.

Problem 13. Given an `unsigned int` (32 bits). Write a C++ program with a function `unsigned int lasttwo(unsigned int)` that return the integer number represented by the last two bits, i.e. if `"00"` return 0, if `"01"` return 1, if `"10"` return 2 and if `"11"` return 3.

Problem 14. Let $n = 0, 1, 2, \ldots$. We define the sequence

$$a_n := \begin{cases} 0 \text{ if the number of 1's in the binary representation} \\ \qquad\qquad \text{of } n \text{ is even} \\ 1 \text{ if the number of 1's in the binary representation} \\ \qquad\qquad \text{of } n \text{ is odd} \end{cases}$$

Write a C++ program that finds this sequence.

Problem 15. Let `v`, `d` be unsigned integers. To find the remainder of integer division `v/d` one use `v%d`. Write a C++ program that uses bitwise operations to do this operation. We assume that `d` is of the form 1, 2, 4, 8, ... i.e. of the form 2^n.

Problem 16. Given an `unsigned int j`. Write a C++ program with a function

```
bool test5(unsigned int j)
```

that tests whether `j` can be divided by 5 without remainder. The bitwise AND must be used.

Problem 17. Given an `unsigned int j`. Write a C++ program with a function

```
bool test3(unsigned int j)
```

that tests whether `j` can be divided by 3 without remainder. The bitwise AND must be used.

Problem 18. Write a C++ program that find the integer part log_2 of an `unsigned int` using bitwise operations.

Problem 19. In the following C++ program we calculate the (floor) of the average of two unsigned integers `x` and `y` as `(x+y)/2`. The result is 352516352 which is obviously wrong. Explain why. Fix the problem using bitwise operations.

```
// average.cpp

#include <iostream>
using namespace std;

int main(void)
{
  unsigned int x = 4000000000;
  unsigned int y = 1000000001;
  unsigned int r1 = (x + y)/2;
  cout << "r1 = " << r1 << endl;
  return 0;
}
```

Problem 20. The bitwise XOR operation `^` has a special property and this can be used directly on *encryption*. Given two arbitrary bitstrings a and b of the same length, then the following expression for bitwise XOR holds

```
(a ^ b) ^ b = a  .
```

In encryption the b would be called the key. Consider a character string, for example "Willi-Hans". Write a C++ program that uses the XOR operation that encrypts and decrypts this character string byte by byte. As a key use

```
unsigned char key = 50;
```

Problem 21. Given a bitstring of length n, where n is even and $n \geq 2$. Test whether the bitstring is alternating or not. For example

```
10101010
01010101
```

are the two alternating bitstrings of length 8.

2.2 Gray Code

Problem 22. An n-bit *Gray code* is a sequence of all the n-bit binary numbers, ordered in such a way that each number differs from its predecessor and its successor by exactly 1 bit and the first and last differ by 1 bit too. Give a 2-bit Grey sequence.

Problem 23. Give two different Gray codes for three bits.

Problem 24. Given a Gray code sequence for $n-1$ bits. Construct a Gray code sequence for n bits using the Gray code sequence for $n-1$ bits.

Problem 25. (i) Consider the traveling saleswoman problem. Given eight cities with the space coordinates

$$(0,0,0), \quad (0,0,1), \quad (0,1,0), \quad (0,1,1),$$

$$(1,0,0), \quad (1,0,1), \quad (1,1,0), \quad (1,1,1).$$

Consider the coordinates as a bitstring. We identify the bitstrings with base 10 numbers, i.e. $000 \to 0$, $001 \to 1$, $010 \to 2$, $011 \to 3$, $100 \to 4$, $101 \to 5$, $110 \to 6$, $111 \to 7$. Thus we set

$$x_0 = (0,0,0), \quad x_1 = (0,0,1), \quad x_2 = (0,1,0), \quad x_3 = (0,1,1)$$

$$x_4 = (1,0,0), \quad x_5 = (1,0,1), \quad x_6 = (1,1,0), \quad x_7 = (1,1,1).$$

The traveling saleswoman does not like her husband. Thus she wants to find the longest route starting at $(0,0,0)$ and returning to $(0,0,0)$ after visiting each city once. Is there more than one solution?
(ii) Consider the traveling salesman problem and the eight cities given in (i). The traveling salesman likes his girlfriend. Thus he wants to find the shortest route starting at $(0,0,0)$ and returning to $(0,0,0)$ after visiting each city once. What is the connection with the *Gray code*? Is there more than one solution?

Problem 26. Extend the previous problem to 16 cities in hyperspace $\mathbb{R}^4$ with the coordinates

$$(0,0,0,0),\quad (0,0,0,1),\quad (0,0,1,0),\quad (0,0,1,1),$$
$$(0,1,0,0),\quad (0,1,0,1),\quad (0,1,1,0),\quad (0,1,1,1),$$
$$(1,0,0,0),\quad (1,0,0,1),\quad (1,0,1,0),\quad (1,0,1,1),$$
$$(1,1,0,0),\quad (1,1,0,1),\quad (1,1,1,0),\quad (1,1,1,1).$$

Problem 27. Give a 4-bit Gray code.

Problem 28. Find a Gray code for 5 bits. Start of with

```
0 0 0 0 0
0 0 0 0 1
0 0 0 1 1
0 0 0 1 0
0 0 1 1 0
0 0 1 1 1
0 0 1 0 1
0 0 1 0 0
0 1 1 0 0
```

Problem 29. The fundamental quantum-dot cellular automata logic device is a three-input *majority logic gate*. The truth table containing all possible input combinations (a, b, c) in Gray code is given by (O is the output)

a	b	c	O
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	0
1	1	0	1
1	1	1	1
1	0	1	1
1	0	0	0

(i) Find the boolean function (sum of products). Can the expression be simplified?
(ii) Show that the majority gate can implement an AND-gate and an OR-gate.
(iii) Can it implement a NOT-gate?

Problem 30. What is the output of the following C++ code?

```
// gray.cpp

#include <iostream>
using namespace std;

unsigned int gray(unsigned int g)
{
g ^= (g >> 16);
g ^= (g >> 8);
g ^= (g >> 4);
g ^= (g >> 2);
g ^= (g >> 1);
return g;
}

int main(void)
{
unsigned int g1 = 0;
unsigned int r1 = gray(g1);
cout << "r1 = " << r1 << endl;

unsigned int g2 = 1;
unsigned int r2 = gray(g2);
cout << "r2 = " << r2 << endl;

unsigned int g3 = 15;
unsigned int r3 = gray(g3);
cout << "r3 = " << r3 << endl;

unsigned int g4 = 256;
unsigned int r4 = gray(g4);
cout << "r4 = " << r4 << endl;

unsigned int g5 = 77;
unsigned int r5 = gray(g5);
cout << "r5 = " << r5 << endl;

return 0;
}
```

2.3 Binary and Arithmetic Operations

Problem 31. Usually we describe arithmetic operation in terms of binary operations, but in some cases it is interesting to consider the reverse problem. Let $a, b \in \mathbb{N}_0$ and $a, b < 2^n$. Thus we can write the binary representation

$$a = \sum_{j=0}^{n-1} a_j 2^j, \qquad a_0, a_1, \ldots, a_{n-1} \in \{0, 1\}$$
$$b = \sum_{j=0}^{n-1} b_j 2^j, \qquad b_0, b_1, \ldots, b_{n-1} \in \{0, 1\}.$$

Now we can define the bitwise operations as functions on whole numbers:

$$\begin{aligned}
NOT(a) : \mathbb{N}_0 \to \mathbb{N}_0, &\qquad NOT(a) = \textstyle\sum_{j=0}^{n-1} NOT2(a_j)2^j,\\
AND(a,b) : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0, &\qquad AND(a,b) = \textstyle\sum_{j=0}^{n-1} AND2(a_j,b_j)2^j,\\
OR(a,b) : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0, &\qquad OR(a,b) = \textstyle\sum_{j=0}^{n-1} OR2(a_j,b_j)2^j,\\
XOR(a,b) : \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0, &\qquad XOR(a,b) = \textstyle\sum_{j=0}^{n-1} XOR2(a_j,b_j)2^j.
\end{aligned}$$

The boolean functions $NOT2$, $AND2$, $OR2$ and $XOR2$ are given by

a_j	b_j	$NOT2(a_j$	$AND2(a_j, b_j)$	$OR2(a_j, b_j)$	$XOR2(a_j, b_j)$
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

In the following use only the arithmetic operations for addition, subtraction, multiplication, quotient and remainder.

1. Give the formula for $NOT(a)$ without referring to the binary representation.
2. Given an algorithm to calculate $AND(a, b)$ without referring to the binary representation.
3. Give a formula for $XOR(a, b)$ in terms of a, b and the AND function.
4. Give a formula for $OR(a, b)$ in terms of a, b and the AND function.
5. Implement bitwise operations on integers using the answers above.

Problem 32. (i) Let $x_j \in \{+1, -1\}$ with $j = 1, 2, \ldots, 9$. Find all solutions of the system of six equations

$$x_1 x_2 x_3 = 1, \quad x_4 x_5 x_6 = 1, \quad x_7 x_8 x_9 = 1,$$

$$x_1x_4x_7 = 1, \quad x_2x_5x_8 = 1, \quad x_3x_6x_9 = -1.$$

Write a C++ program that runs of all 2^9 combinations to find the solutions.
(ii) Let $y_j \in \{0, 1\}$. Find all solutions of the system of six equations

$$y_1 \oplus y_2 \oplus y_3 = 1, \quad y_4 \oplus y_5 \oplus y_6 = 1, \quad y_7 \oplus y_8 \oplus y_9 = 1,$$

$$y_1 \oplus y_4 \oplus y_7 = 1, \quad y_2 \oplus y_5 \oplus y_8 = 1, \quad y_3 \oplus y_6 \oplus y_9 = -1$$

where $\oplus$ denotes the XOR-operation.

2.4 Theory

Problem 33. Let $n \geq 2$ and even. How many bitstrings of length n can one form with $n/2$ 0's and $n/2$ 1's. For $n = 4$ write down all of them in lexicographical order.

Problem 34. Consider a binary string S of length $n \geq 1$ with symbols 1 and 0. We define $\Delta(S)$ as the number of 1's of S minus the number of 0's. For example, $\Delta(\text{“}1001001\text{”}) = -1$. We call a string S balanced if every substring T of consecutive symbols of S has $-2 \leq \Delta(T) \leq 2$. Thus “1001001” is not balanced, since it contains the substring “00100”. The string 01010101 is balanced. Let b_n be the number of balanced strings of length n. Obviously we have $b_1 = 2$ with the strings “0” and “1” and $b_2 = 4$ with the strings “00”, “01”, “10” and “11”. Find a recursion relation for b_n.

Problem 35. Let

$$\alpha = 1 + a_1x + a_2x^2 + \cdots$$

be a formal power series with coefficients in the field of two elements (characteristic 2). We define

$$a_n := \begin{cases} 1 \text{ if every block of zeros in the binary expansion} \\ \quad \text{of } n \text{ has an even number of zeros in the block} \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

For example, $a_{11} = 0$ since $11 = 1011_2$ and $a_{36} = 1$ since $36 = 100100_2$. Show that $\alpha^3 + x\alpha + 1 = 0$.

Problem 36. Consider two bitstrings of length N,

$$\mathbf{b} = b_0b_1b_2 \ldots b_{N-1}, \qquad \mathbf{c} = c_0c_1c_2 \ldots c_{N-1}$$

and the map

$$c_0 = b_0$$

$$
\begin{aligned}
c_1 &= b_0 \oplus b_1 \\
c_2 &= b_0 \oplus b_1 \oplus b_2 \\
&\vdots \\
c_{N-1} &= b_0 \oplus b_1 \oplus \cdots \oplus b_{N-1}
\end{aligned}
$$

(i) Let $\mathbf{b} = b_0 b_1 b_2 b_3 = 1011$, i.e. $N = 4$. Calculate $\mathbf{c}$.
(ii) Find the inverse map if it exists.

Problem 37. We consider multiplication of two two-binary numbers $a_1 a_0$ and $b_1 b_0$. For example, if $a_1 a_0 = 10$ (2 in decimal) and $b_1 b_0 = 11$ three in decimal, then we find for the product (output) $O_3 O_2 O_1 O_0 = 0110$ (6 in decimal). Find all possible products and put them into a function table.

Problem 38. The algebraic *Reed-Muller expansion* of boolean functions is one of the fundamental approaches in the design of digital systems, especially when dealing with the VLSI technology. This algebraic representation is useful in error detection and error correction models. Let $f(x_1, x_2, \ldots, x_n)$ be a boolean function. We define with respect to x_j $(j = 1, 2, \ldots, n)$

$$
\begin{aligned}
f_{x_j}(x) &:= f(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n) \\
f_{\bar{x}_j}(x) &:= f(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n) \\
\frac{\partial f}{\partial x_j} &:= f_{x_j}(x) \oplus f_{\bar{x}_j}(x)
\end{aligned}
$$

as the positive cofactors of f, negative cofactor of f, and the boolean derivative of f. Then the Reed-Muller expansion (also called Davio expansion) is given by

$$
f = f_{\bar{x}_j} \oplus \left(x_j \cdot \frac{\partial f}{\partial x_j} \right)
$$

where $\oplus$ is the XOR operation and $\cdot$ is the AND operation. The complement of the variable is denoted by an overbar, that is $\bar{x} = 1 - x$. Note that $(a \oplus b) \cdot c \neq a \oplus (b \cdot c)$ in general, for example for $a = 1$, $b = 0$, $c = 0$. Consider the sum-of-product form of the boolean function

$$
f(x_1, x_2, x_3, x_4) = \bar{x}_1 \cdot x_3 + \bar{x}_1 \cdot x_2 \cdot x_4 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 + x_1 \cdot \bar{x}_3 \cdot \bar{x}_4.
$$

Apply the Reed-Muller expansion to find a simpler expression.

Problem 39. Let $f(x_1, x_2, \ldots, x_n)$ be a boolean function of n variables. *Shannon's expansion* is given by

$$
f(x_1, x_2, \ldots, x_n) = x_1 \cdot f_1(1, x_2, \ldots, x_n) + \bar{x}_1 \cdot f(0, x_2, \ldots, x_n) \qquad (1)
$$

and

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 \cdot f(0, x_2, \ldots, x_n) \oplus x_1 \cdot f(1, x_2, \ldots, x_n). \quad (2)$$

Apply these two expansions to the boolean function

$$f(x_1, x_2, x_3) = (x_1 + x_2) \cdot (x_1 + x_3).$$

Discuss.

Problem 40. A *J-K flip-flop* can memorize a single bit of information. The next state $Q(t+1)$ of a J-K flip lop is characterized as a function of both the present state $Q(t)$ and the present two inputs $J(t)$ and $K(t)$. The truth table for the J-K flip flop is

$J(t)$	$K(t)$	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Find the minterm expression and simplify it.

Problem 41. Consider the set $\{+1, -1\}$ and multiplication. Then we have the group table

$\cdot$	+1	-1
+1	+1	-1
-1	-1	+1

The neutral element of the group is +1. Consider now the set $\{0, 1\}$ and the XOR operation. Then we have the group table

$\oplus$	0	1
0	0	1
1	1	0

The neutral element of the group is 0. Show that the two group are isomorphic.

Problem 42. The *Cantor sequence* is constructed as follows. Given the natural numbers

$$n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots$$

We write them in ternary notation as

$$0, 1, 2, 10, 11, 12, 20, 21, 22, 100, \ldots$$

Then the Cantor sequence b_n $(n = 0, 1, 2, \ldots)$ is defined as: if n in ternary has only 0s and 2s, then $b_n = 1$, otherwise we set $b_n = 0$. The first 9 terms of the sequence are given by

$$1,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1, \ldots$$

Write a C++ program that generates this sequence. Is the sequence chaotic?

Problem 43. The *Thue-Morse sequence* is defined as follows. Let $V = \{\, 0, 1 \,\}$, $s_0 = 0$ (intial value of the sequence) and

$$p_1 : 0 \to 01, \qquad p_2 : 1 \to 10.$$

Find the first five elements in the construction of the Thue-Morse sequence. Write a C++ program that generates the sequence. Use the bitset class.

Problem 44. Consider the map

$$f(x) = 2x \bmod 1.$$

We can associate with each point in $[0, 1]$ an itinerary (an infinite sequence of 0's and 1's) on which the shift map represents the action of f. Show that all points in the subinterval

$$[(k-1)2^{-n}, k^{2^{-n}}], \qquad 1 \le k \le 2^n$$

are represented by a finite sequence of n symbols.

Problem 45. A boolean function $f : \{\, 0, 1 \,\}^{\otimes n} \to \{\, 0, 1 \,\}^{\otimes m}$ $(m \ge n)$ is periodic with period p with respect to bitwise modulo 2 addition, i.e. for all x we have

$$f(x) = f(x + p).$$

Give an example of such a periodic boolean function.

Problem 46. Let $\mathbf{x} = x_1 x_2 \ldots x_n$, $\mathbf{y} = y_1 y_2 \ldots y_n$, $\mathbf{z} = z_1 z_2 \ldots z_n$ be three n-bit binary strings. Let $\oplus$ denote bitwise addition modulo 2, i.e.

$$\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \Rightarrow \text{for all } k,\ z_k = x_k + y_k \bmod 2.$$

We define a scalar product of $\mathbf{x}$ and $\mathbf{y}$ by

$$\mathbf{x} \bullet \mathbf{y} := (x_1 \cdot y_1) + (x_2 \cdot y_2) + \cdots + (x_n \cdot y_n) \bmod 2.$$

Show that this binary scalar product $\bullet$ is distributive over bitwise modulo 2 addition $\oplus$, i.e.

$$(\mathbf{x} \oplus \mathbf{y} \bullet \mathbf{z} = (\mathbf{x} \bullet \mathbf{z}) \oplus (\mathbf{y} \bullet \mathbf{z}).$$

Problem 47. Consider the full adder given by the truth table

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Implement the full adder using only NAND-gates.

Problem 48. The Cantor series approximation is defined as follows. For arbitrary chosen integers $n_1, n_2, \ldots$ (equal or larger than 2), we can approximate any real number r_0 as follows

$$\begin{aligned} x_j &= \text{integer part}(r_j), \quad j = 0, 1, 2, \ldots \\ r_{j+1} &= (r_j - x_j) n_j \end{aligned}$$

and

$$r_0 \approx x_0 + \sum_{j=1}^{N} \frac{x_j}{n_1 n_2 \cdots n_j}.$$

The approximation error is

$$E_n = \frac{1}{n_1 n_2 \cdots n_N}.$$

Apply this approximation to $r_0 = 2/3$ and the golden mean number with $n_j = 2$ for all j and $N = 4$. Thus the x_j form a bit sequence.

Problem 49. Let

$$\Sigma_2 := \{\, \mathbf{s} := (s_0 s_1 s_2 \ldots) \,:\, s_j = 0 \text{ or } 1 \,\}.$$

Σ_2 is known as sequence space on the symbols 0 and 1. If we define the distance between the two sequences $\mathbf{s}$ and $\mathbf{t}$ by

$$d[\mathbf{s}, \mathbf{t}] = \sum_{j=0}^{\infty} \frac{|s_j - t_j|}{2^j}$$

then Σ_2 is a metric space.
(i) Let $\mathbf{s} = (1111\ldots)$, $\mathbf{t} = (0000\ldots)$. Find the distance.
(ii) Consider the periodic sequences

$$\mathbf{s} = (010101\ldots), \qquad \mathbf{t} = (101010\ldots).$$

Find the distance.
(iii) A dynamics is given by the *shift map* $\sigma : \Sigma_2 \to \Sigma_2$ defined by

$$\sigma(s_0 s_1 s_2 \ldots) := (s_1 s_2 s_3 \ldots).$$

Let $\mathbf{s} = (1000\cdots)$ and $\mathbf{t} = \sigma(\mathbf{s})$. Find the distance.

Problem 50. Periodic points are identified with exactly repeating sequences

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

How many periodic points of period n there are?

Problem 51. Show that

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

using (i) truth tables and (ii) properties of boolean algebra (with $a+1 = 1$).

Problem 52. (i) Given a digital circuit with 4 input lines and four output lines. The output is the two's complement of the input. Give the truth table.
(ii) Find the disjunctive normal form.
(iii) Construct a logic circuit using a PAL.
(iv) Write a VHDL program tha simulates this circuit.

Problem 53. The genetic code vector space is presented by the *Galois field* of four bases ($GF(4)$). We consider the invertible map $f : \{A, C, G, T\} \to \{(a_j, a_{j+1})\}$ from the base set to the set of binary duplets a_j, a_{j+1}, where $f(X) = (a_j, a_{j+1})$ and $a_j \in \{0, 1\}$. Since A maps to T, C maps to G we introduce the map

$$f(A) = (0,0), \quad f(C) = (1,0), \quad f(G) = (0,1), \quad f(T) = (1,1).$$

We also write (0,0) etc as bitstrings `"00"`. Using the NOT-operation we have $\overline{00} = 11$, $\overline{01} = 10$. Thus the sum of binary digits corresponding to DNA complementary basis is always $(1, 1)$. Therefore we have eight ordered base sets, namely

$$\{G, A, T, C\}, \quad \{G, T, A, C\}, \quad \{C, A, T, G\}, \quad \{C, T, A, G\}$$

$$\{A, C, G, T\}, \quad \{A, G, C, T\}, \quad \{T, C, G, A\}, \quad \{T, G, C, A\}$$

Write a C++ program using the `bitset` class that converts a given DNA sequence, for example "ATGCAATTCTCGCTA", into the corresponding bitstring.

Problem 54. Solve the following Max-SAT problem with five clauses and four variables x_1, x_2, x_3, x_4

$$\begin{aligned} f_1(x_1, x_2.x_3, x_4) &= x_1 \vee \neg x_2 \\ f_2(x_1, x_2, x_3, x_4) &= \neg x_1 \vee x_3 \vee \neg x_4 \\ f_3(x_1, x_2, x_3, x_4) &= \neg x_1 \vee \neg x_2 \\ f_4(x_1, x_2, x_3, x_4) &= x_1 \vee \neg x_3 \vee x_4 \\ f_5(x_1, x_2, x_3, x_4) &= x_2 \vee x_3 \vee \neg x_4 \end{aligned}$$

using backtracking branch-and-bound, where $\neg$ is the NOT and $\vee$ is the OR.

Problem 55. Let ℓ and m be positive integers satisfying $1 \leq \ell < m$. Let

$$\mathbf{X} := X_{-m+1}, \ldots, X_{-1}, X_0, X_1, X_2, \ldots$$

be a binary sequence satisfying the recursion relation

$$X_k = X_{k-\ell} \oplus X_{k-m}$$

for $k = 1, 2, \ldots$. Here $\oplus$ denotes addition modulo 2. Any such sequence will be called a linear-feedback shift sequence. For $k \geq 0$ the m-tuple

$$[\mathbf{X}] := (X_{k-m+1}, \ldots, X_{k-1}, X_k)$$

will be called the state of $\mathbf{X}$ at time k. Let $\ell = 1$, $m = 2$, $X_{-1} = 1$, $X_0 = 1$. Find the sequence X_1, X_2, ... Write a C++ pogram that implements this sequence.

Problem 56. A Boolean function $f : \{0,1\}^2 \to \{0,1\}$ is constant if for every $x \in \{0,1\}^2$ we have $f(x) = c$, where $c \in \{0,1\}$. A Boolean function is balanced if

$$|\{x \in \{0,1\}^2 \, : \, f(x) = 0\}| = |\{x \in \{0,1\}^2 \, : \, f(x) = 1\}|$$

i.e. f maps to an equal number of zeros and ones. Find all Boolean functions from $\{0,1\}^2$ to $\{0,1\}$ which are either constant or balanced. Determine which of these functions are separable, i.e. for which function can we find a hyperplane which separates the inputs x which give $f(x) = 1$ and the inputs y which give $f(y) = 0$.

Problem 57. The average information gain when one of a mutually exclusive set of events with probabilities $p_0, \ldots, p_{n-1}$ occurs is defined as the *Shannon entropy*

$$H(p_0, p_1, \ldots, p_{n-1}) := -\sum_{j=0}^{n-1} p_j \log(p_j)$$

with the convention $0 \log(0) = 0$. Consider an information source that transmits words composed out of letters of an alphabet of eight elements. If all letters have equal probability, $p_j = 1/8$, $j = 0, 1, \ldots, 7$, to appear per transmitted letter, a compact way of representing the letters in binary notation is the 3-bit code

$$\begin{aligned} 0th &\to 000, \\ 1st &\to 001, \\ &\vdots \\ 7th &\to 111. \end{aligned}$$

Find the Shannon entropy.

Problem 58. Show that the XOR-gate can be built from 4 NAND-gates. Let A_1, A_2 be the inputs and O the output.

Problem 59. Consider the function $f : [-1, 1] \times [-1, 1] \to \mathbb{R}$

$$f(x_1, x_2) = x_1 x_2.$$

We want to find the maxima of the (fitness) function f in the given domain $[-1, 1] \times [-1, 1]$. Consider the two bitstrings

```
b1 = "0000000011111111"     b2 = "1010101010101010"
```

where the 8 bits on the right-hand side belong to x_1 and the 8 bits on the left-hand side belong to x_2. Find the value of the fitness function for the two bitstring. Apply the NOT-operation to the two bitstrings to obtain the new bitstrings $b3 = NOT(b1)$ and $b4 = NOT(b2)$. Select the two fittest of the four bitstrings for survival.

Problem 60. Consider the domain $[-1, 1] \times [-1, 1]$ in $\mathbb{R}^2$ and the bitstring (16 bits)

```
0101010100001111
```

The 8 bits on the right-hand side belong to x and the 8 bits on the left-hand side belong to y (counting from right to left starting at 0). Find the real

values x^* and y^* for this bitstring. Let $f : [-1, 1] \times [-1, 1] \to \mathbb{R}$ be given by

$$f(x, y) = x^4 + y^4.$$

Find the value $f(x^*, y^*)$. How far is this value from the minimum of f in the given domain?

Problem 61. Show that the *full-adder* can be built with nine NAND-gates. Give the circuit. Describe the circuit using *multiexpression programming*.

Problem 62. Let $a, b \in \{0, 1\}$ and

```
0: a
1: b
2: NOT 0
3: OR  1,2
```

Give the output.

Problem 63. Let $a, b, c, d \in \{0, 1\}$ and

```
0: a
1: b
2: c
3: d
4: XOR 0,2
5: XOR 1,3
6: OR  4,5
```

Give the output.

Problem 64. Let $a, b \in \{0, 1\}$. Consider the sequence of expressions in multiexpression programming

```
0: a
1: b
2: NOT 0
3: OR 1,2
```

Give the output at 3.

Problem 65. Let $a, b, c, d \in \{0, 1\}$. Consider the sequence of expressions in multiexpression programming

```
0: a
1: b
2: c
3: d
4: XOR 0,2
5: XOR 1,3
6: OR  4,5
```

Give the output at 6.

Problem 66. Express the majority gate

```
I_1 I_2 I_3  O
 0   0   0   0
 0   0   1   0
 0   1   0   0
 0   1   1   1
 1   0   0   0
 1   0   1   1
 1   1   0   1
 1   1   1   1
```

with multiexpression programming.

Problem 67. Build the 4 bit magnitude comparator using only NAND gates.

Problem 68. Build the 8 bit priority encoder using only NAND gates.

Problem 69. (i) Consider a bit-string of length n counting the bit-string from right to left and starting counting from 0. Write a C++ program using the `bitset` class that selects at random a bit and then the bits on the left and on the right are switched around and the selected bit stays the same. This means all bits except the selected bit are negated. For example consider the bitstring `10101110` and we select bit number 2, then we have `01010101`.

(ii) Assume that all bits are set in the bitstring. Can one find a sequence of this operation so that all bits are off, i.e. set to 0.

Problem 70. In C++ the data type `double` consists of 64 bits. Convert this bitstring of a `double` into a character array `char a[8]` counting from left to right in the bitstring of the `double`.

Problem 71. The data type `float` consists of 32 bits. Thus the number of possible bitstrings is

$$2^{32} = 68719476736$$

Apply the bitset class of C++ to generate all these bitstrings and convert them into the corresponding `float` or `NaN` (not a number).

Problem 72. Let

$$\sigma_{0,0}, \sigma_{1,0}, \sigma_{-1,0}, \sigma_{0,1}, \sigma_{0,-1} \in \{0, 1\}.$$

Consider the Hamiltonian

$$H = \sigma_{0,1} \oplus \sigma_{0,0} \oplus \sigma_{-1,0} + \sigma_{-1,0} \oplus \sigma_{0,0} \oplus \sigma_{1,0}$$

where $\oplus$ denotes the XOR-operation and $+$ is the arithmetic addition. Find the energy levels of H by running through all possible bistrings. There are 2^5 bitstrings of length 5. Write a C++ program using the `bitset` class that finds all the levels.

Problem 73. Show that the NAND-operation is not associative in general, i.e. $N(N(a, b), c) \neq N(a, N(b, c))$. Consider the case $a = 0$, $b = 0$, $c = 1$.

Problem 74. Consider the map

$$x_{t+1} = 2x_t \mod 1$$

with $x_0 \in [0, 1)$ to find the binary expansion of the rational number $x_0 = 3/13$.

Problem 75. Let $\mathbb{F}_2$ be the field with the elements 0 and 1. Consider the vector $\mathbf{v} := (v_0, v_1, \ldots, v_{m-1})$ from the vector space $\mathbb{F}_2^m$. Then the complement of the vector $\mathbf{v}$ is defined as

$$\bar{\mathbf{v}} := (1 - v_0, 1 - v_1, \ldots, 1 - v_{m-1}).$$

We define a subset of vectors $W_n \subseteq \mathbb{F}_2^{2^n}$ as follows: We set $W_0 := \{(0)\}$ and for $j \geq 0$ the set W_{j+1} consists of all vectors given by $(\mathbf{w}, \mathbf{w})$ and $(\mathbf{w}, \bar{\mathbf{w}})$, where $\mathbf{w}$ is any vector from W_j. Therefore W_{j+1} has twice as many vectors as the set W_j. One calls W_n the set of *Hadamard vectors*. The length of the Hadamard vectors in W_n is 2^n.
(i) Let $W_1 = \{(0, 0), (0, 1)\}$. Find W_2.
(ii) Write a C++ program using the `bitset` class to generate the Hadamard vectors.

Problem 76. Let n be an odd number and $n \geq 3$. Write a C++ program using the bitset class that parses trough all bitstrings of length n. If the number of 0's is larger then the number of 1's the output should be 0 and the number of 0's is smaller then the number of 1's the output should be 1. Then write down the truth table (majority gate).

Problem 77. Consider the lattice

```
        (0,1)
(-1,0)  (0,0)  (1,0)
       (-1,0)
```

and the Hamiltonian

$$H = \sigma_{0,1} \oplus \sigma_{0,0} \oplus \sigma_{-1,0} + \sigma_{-1,0} \oplus \sigma_{0,0} \oplus \sigma_{1,0}$$

where $\sigma_{0,0}, \sigma_{1,0}, \sigma_{-1,0}, \sigma_{0,1}, \sigma_{0,-1} \in \{0,1\}$, $\oplus$ is the XOR-operation and $+$ the arithmetic addition. Find the enery levels of $\hat{H}$ by running through all possible bitstrings. There are $2^5 = 32$ bitstrings of length 5. Write a C++ programs that finds the energy level using the `bitset` class of C++.

Problem 78. Let $x \in \{0,1\}$. Consider the boolean functions $(f_j : \{0,1\} \to \{0,1\})$

$$f_1(x) = x \oplus x, \quad f_2(x) = x \cdot x, \quad f_3(x) = x + x$$

where $\oplus$ is the XOR-operation, $\cdot$ is the AND-operation and $+$ is the OR-operation. Find the fixed points of f_1, f_2 and f_3.

Problem 79. Consider the boolean function $\mathbf{f} : \{0,1\}^2 \to \{0,1\}^2$

$$f_1(x,y) = y \oplus y, \qquad f_2(x,y) = x \oplus x.$$

(i) Find the fixed points of the boolean function.
(ii) Start with $(x,y) = (1,1)$ and iterate the boolean function. Discuss.
(iii) Start with $(x,y) = (1,0)$ and iterate the boolean function. Discuss.
(iv) Start with $(x,y) = (0,1)$ and iterate the boolean function. Discuss.

Problem 80. Consider the boolean function $\mathbf{f} : \{0,1\}^2 \to \{0,1\}^2$

$$f_1(x,y) = x \oplus y, \qquad f_2(x,y) = x \oplus y.$$

(i) Find the fixed points of the boolean function.
(ii) Start with $(x,y) = (1,0)$ and iterate the boolean function at least twice. Discuss.

(iii) Start with $(x, y) = (0, 1)$ and iterate the boolean function at least twice. Discuss.
(iv) Start with $(x, y) = (1, 1)$ and iterate the boolean function. Discuss.

Problem 81. Given the boolean function $f : \{0, 1\}^3 \to \{0, 1\}$ as truth table

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

Consider the unit cube in $\mathbb{R}^3$ with vertices (corner points) $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$, $(1, 1, 0)$, $(1, 1, 1)$. Find the plane in $\mathbb{R}^3$ that separates the 0 from the 1's.

Problem 82. Consider the boolean function $\mathbf{f} : \{0, 1\}^3 \to \{0, 1\}^3$

$$f_1(x, y, z) = y \oplus y \oplus y, \qquad f_2(x, y, z) = z \oplus z \oplus z, \qquad f_3(x, y, z) = x \oplus x \oplus x.$$

(i) Find the fixed points of the boolean function.
(ii) Start with $(x, y, z) = (1, 0, 1)$ and iterate the boolean function. Discuss.
(iii) Start with $(x, y, z) = (1, 0, 0)$ and iterate the boolean function. Discuss.
(iv) Start with $(x, y, z) = (0, 1, 0)$ and iterate the boolean function. Discuss.

Problem 83. The NAND-gate is given by

```
i_1  i_2  o
 0    0   1
 0    1   1
 1    0   1
 1    1   1
```

where `i_1`, `i_2` are the inputs and `o` is the output. The XNOR-gate is given by

```
i_1  i_2  o
 0    0   1
 0    1   0
 1    0   0
 1    1   1
```

where `i_1`, `i_2` are the inputs and `o` is the output. Show that the XNOR-gate can be build with five NAND-gates.

Problem 84. Let $n \geq 1$. Consider the boolean function $f : \{0,1\}^n \to \{0,1\}$. Then the *boolean derivatives* are defined as

$$\left.\frac{\partial f}{\partial x_j}\right|_x := f(x_1, \ldots, x_j, \ldots, x_n) \oplus f(x_1, \ldots, \bar{x}_j, \ldots, x_n)$$

for $j = 1, \ldots, n$.
(i) Consider the boolen function $f : \{0,1\}^2 \to \{0,1\}$ given by

$$f(x_1, x_2) = x_1 + x_2.$$

Find the boolean derivatives $\partial f/\partial x_1$, $\partial f/\partial x_2$.
(ii) Consider the boolen function $f : \{0,1\}^2 \to \{0,1\}$ given by

$$f(x_1, x_2) = x_1 \cdot x_2.$$

Find the boolean derivatives $\partial f/\partial x_1$, $\partial f/\partial x_2$.
(iii) Consider the boolean function $f : \{0,1\}^4 \to \{0,1\}$

$$f(x_1, x_2, x_3, x_4) = x_1 + x_2 \oplus x_3 \cdot x_4.$$

Find the boolean derivatives $\partial f/\partial x_1$, $\partial f/\partial x_2$, $\partial f/\partial x_3$, $\partial f/\partial x_4$.

Problem 85. (i) The binary *Thue-Morse sequence* is generated by the substitutions

$$0 \mapsto 01, \qquad 1 \mapsto 10.$$

Start with 0 and give the sequence for the first four steps. Give a C++ implementation.
(ii) Study the binary sequence given by the substitutions

$$0 \mapsto 11, \quad 1 \mapsto 10$$

for the first four steps starting with 1. Give a C++ implementation.
(iii) The Rudin-Shapiro sequence is generated by the two-digit substitutions

$$00 \mapsto 0001, \quad 01 \mapsto 0010, \quad 10 \mapsto 1101, \quad 11 \mapsto 1110.$$

Start with 00 and give the sequence after three substitutions. Give a C++ implementation.

Problem 86. Consider the NAND gate. Show that the NOT gate can be build with one NAND gate. Show that the AND gate can be build with

two NAND gates. Show that the OR gate can be build with three NAND gates. Show that the NOR gate can be build with four NAND gates. Show that the XOR gate can be build with four NAND gates. Show that the XNOR gate can be build with five NAND gates.

Problem 87. Consider the unit cube with the 8 corner points $jk\ell$ ($j, k, \ell \in \{0, 1\}$)

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111.$$

Let $x_{jk\ell} \in \{0, 1\}$. Each corner point has three nearest neighbours. Find all the energy levels for

$$\begin{aligned} E = {} & x_{000} \oplus x_{001} \oplus x_{010} \oplus x_{100} + x_{001} \oplus x_{000} \oplus x_{011} \oplus x_{101} \\ & + x_{010} \oplus x_{000} \oplus x_{011} \oplus x_{110} + x_{011} \oplus x_{001} \oplus x_{010} \oplus x_{111} \\ & + x_{100} \oplus x_{000} \oplus x_{110} \oplus x_{101} + x_{101} \oplus x_{001} \oplus x_{100} \oplus x_{111} \\ & + x_{110} \oplus x_{111} \oplus x_{100} \oplus x_{010} + x_{111} \oplus x_{011} \oplus x_{110} \oplus x_{101} \end{aligned}$$

where $\oplus$ is the XOR-operation and $+$ is the arithmetic plus. Find the energy level for all possible $2^8 = 256$ configurations.

Problem 88. (i) Let $s_1(0), s_2(0), s_3(0) \in \{+1, -1\}$. Study the time-evolution ($t = 01, 2, \ldots$) 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 ($t = 01, 2, \ldots$) 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?

Problem 89. 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?

Problem 90. The Cantor series approximation is defined as follows. For arbitrary chosen integers $n_1, n_2, \ldots$ (equal or larger than 2), we can approximate any real number r_0 as follows

$$\begin{aligned} x_j &= \text{integer part}(r_j), \quad j = 0, 1, 2, \ldots \\ r_{j+1} &= (r_j - x_j) n_j \end{aligned}$$

and

$$r_0 \approx x_0 + \sum_{j=1}^{N} \frac{x_j}{n_1 n_2 \cdots n_j}.$$

The approximation error is

$$E_n = \frac{1}{n_1 n_2 \cdots n_N}.$$

Apply this approximation to $r_0 = 2/3$ and the golden mean number with $n_j = 2$ for all j and $N = 4$.

Problem 91. Show that the *full-adder* can be built with nine NAND-gates. Give the circuit. Describe the circuit using *multiexpression programming*.

Problem 92. A boolean function $f : \{0, 1\}^{\otimes n} \to \{0, 1\}^{\otimes m}$ ($m \geq n$) is periodic with period p with respect to bitwise modulo 2 addition, i.e. for all x we have

$$f(x) = f(x + p).$$

Give an example of such a periodic boolean function.

Problem 93. Let $\mathbf{x} = x_1x_2\dots x_n$, $\mathbf{y} = y_1y_2\dots y_n$, $\mathbf{z} = z_1z_2\dots z_n$ be three n-bit binary strings. Let $\oplus$ denote bitwise addition modulo 2, i.e.

$$\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \Rightarrow \text{for all } k,\ z_k = x_k + y_k \bmod 2$$

We define a scalar product of $\mathbf{x}$ and $\mathbf{y}$ by

$$\mathbf{x} \bullet \mathbf{y} := (x_1 \cdot y_1) + (x_2 \cdot y_2) + \cdots + (x_n \cdot y_n) \bmod 2.$$

Show that this binary scalar product $\bullet$ is distributive over bitwise modulo 2 addition $\oplus$, i.e.

$$(\mathbf{x} \oplus \mathbf{y} \bullet \mathbf{z} = (\mathbf{x} \bullet \mathbf{z}) \oplus (\mathbf{y} \bullet \mathbf{z}).$$

Problem 94. Let $x \in \{\,0, 1\,\}$. Calculate

$$f_1(x,y,z) = (x{\cdot}(y{+}z)), \quad f_2(x,y,z) = (x{\cdot}(y{\oplus}z), \quad f_3(x,y,z) = (x{+}(y{\oplus}z)).$$

Problem 95. Let $n \in \mathbb{N}_0$. Find the number of binary words consisting of $(n+1)$ 0's and n 1's that can be formed such that the number of 0's in every subword, when read from far left to righ, is greater than the number of 1's in it. For $n = 0$ we have 0. For $n = 1$ we have 001.

Problem 96. Consider the unit cube with the 8 vertices

$$(0,0,0),\ \ (0,0,1),\ \ (0,1,0),\ \ (0,1,1)$$

$$(1,0,0),\ \ (1,0,1),\ \ (1,1,0),\ \ (1,1,1).$$

Find a *Hamilton cycle* starting from $(0,0,0)$.

Problem 97. Let $x, y, a, b \in \{0,1\}$. Consider the function

$$f(x,y,a,b) = \begin{cases} 1 & \text{if} \quad x \cdot y = a \oplus b \\ 0 & \text{otherwise} \end{cases}$$

where $\cdot$ is the AND operation and $\oplus$ is the XOR operation. Write a C++ program that finds the function f for all 16 possible inputs utilizing the `bitset` class of the standard template library. The function plays a role for the CHSH game.

Problem 98. Let $a, b, c \in \{0,1\}$ and $\cdot$ be the AND-operation, $+$ be the OR-operation, $\oplus$ be the XOR-operation.

(i) Find all solutions of

$$a \cdot b \cdot c = a + b + c.$$

(ii) Find all solutions of

$$a \cdot b \cdot c = a \oplus b \oplus c.$$

(iii) Find all solutions of

$$a + b + c = a \oplus b \oplus c.$$

Chapter 3

Binary Matrices

A binary matrix (also called $(0,1)$ matrix) is an $m \times n$ matrix in which each entry is either zero or one. For example

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

is a 2×4 binary matrix. The number of $m \times n$ binary matrices is obviously 2^{mn}. In most cases operations on binary matrices are defined in terms of modular arithmetic mod 2. This means the elements are treated as elements of the Galois field $GF(2) = \mathbb{Z}_2$. Thus $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. For example in standard addition we have, for example

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$$

but in modular arithmetic mod 2 we have

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

If we have two binary matrices of the same size, then the Hadamard product (also called Schur product) would be an entry-wise AND-operation. For example

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \bullet \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

The $n \times n$ permutation matrices are binary matrices, all of whose columns and rows have each exactly one nonzero element. An adjacent matrix in

graph theory is a matrix whose rows and columns represent the vertices and whose entries represent the edges of the graph. The adjaceny matrix of a simple undirected graph is a binary symmetric matrix with 0 diagonal.

Problem 1. (i) Find all 2×2 binary matrices.
(ii) Find the determinant of all these matrices with the underlying field $\mathbb{R}$.
(iii) In modular arithmetic mod 2 we define for a binary 2×2 matrix

$$\det 2(A) := (a_{11} \cdot a_{22}) \oplus (a_{12} \cdot a_{21})$$

where $\cdot$ is the AND-operation and $\oplus$ the XOR operation. Find $\det 2(A)$ for all 2×2 binary matrices.

Problem 2. Write a C++ program that generates all $m \times n$ binary matrices.

Problem 3. Write a C++ program that finds the determinant of an $n \times n$ binary matrix over the field $\mathbb{Z}_2$.

Problem 4. Find all binary 2×2 matrices with no adjacent 1s (in either columns or rows).

Problem 5. A special case of binary matrices are the *permutation matrices*. An $n \times n$ permutation matrix contains exactly one 1 in each row and column. The other entries are 0. Each permutation matix has an inverse. The determinant of a permutation matrix is either $+1$ or -1. The permutation matrices form a group under matrix multiplication with the underlying field $\mathbb{R}$. Find all 3×3 permutation matrices. Calculate the matrix products of these matrices with the underlying field $\mathbb{R}$ and with the underlying field $\mathbb{Z}_2$.

Problem 6. Consider the bitstrings 00, 01, 10, 11 and the XOR-operation $\oplus$. We define

$$(a_1 a_0) \oplus (b_1 b_0) = (a_1 \oplus b_1)(a_0 b_0).$$

Thus we have the (group) table

$\oplus$	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

Consider the 2×2 matrices

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

and matrix multiplication. Find the group table. Show that the two groups are isomorphic.

Problem 7. Consider the bistrings 00, 01, 10, 11 and the rotation operation with

$$R(00) = 00, \quad R(01) = 10, \quad R(10) = 01, \quad R(11) = 11.$$

Now the bistrings can be mapped into 2×2 matrices

$$00 \to \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2, \quad 01 \to \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad 10 \to \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad 11 \to \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}.$$

How can the rotation operation be implemented using an invertible 2×2 matrix.

Problem 8. Extend the previous problem to bitstrings of length 3, i.e. 000, 001, 010, 011, 100, 101, 110, 111 and the rotation

$$R(000) = 000, \quad R(001) = 100, \quad R(010) = 010, \quad R(011) = 101$$

$$R(100) = 010, \quad R(101) = 101, \quad R(110) = 011, \quad R(111) = 111$$

The map into the 3×3 diagonal matrices

$$000 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = I_3, \quad 001 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{pmatrix}$$

$$010 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad 011 \to \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

$$100 \to \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = I_3, \quad 101 \to \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$110 \to \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad 111 \to \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

Problem 9. Consider the Pauli spin matrices

$$\sigma_0 = I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

where we include the 2×2 identity matrix. We can associate binary numbers with a Pauli spin matrix (including the identity matrix I_2) σ_j $(j = 0, 1, 2, 3)$ via a two-dimensional binary vector $\mathbf{r}(\sigma_j)$ with

$$\mathbf{r}(\sigma_0) = (0\,0), \quad \mathbf{r}(\sigma_1) = (1\,0), \quad \mathbf{r}(\sigma_2) = (1\,1), \quad \mathbf{r}(\sigma_3) = (0\,1).$$

The first and second entries of this bit vector are written as $r_1(\sigma_j)$ and $r_2(\sigma_j)$, respectively. Show that given the binary representation of a Pauli matrix (including the identity matrix) we can recover the matrix via

$$\sigma_j = i^{r_1(\sigma_j)r_2(\sigma_j)}\sigma_1^{r_1(\sigma_j)}\sigma_3^{r_2(\sigma_j)}.$$

Problem 10. Given two binary matrices A and B. Show that the *Kronecker product* $A \otimes B$ is also a binary matrix.

Problem 11. Given two binary matrices A and B. Show that the *direct sum* $A \oplus B$ is also a binary matrix.

Problem 12. For a 2×2 binary matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad a_{jk} \in \{\,0,\,1\,\}$$

we define the determinant as

$$\det A = (a_{11} \cdot a_{22}) \oplus (a_{12} \cdot a_{21})$$

where $\cdot$ is the AND-operation and $\oplus$ is the XOR-operation.
(i) Find the determinant for the following 2×2 matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

(ii) Find the determinant for the following 2×2 matrices

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Problem 13. The determinant of a 3×3 matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

is given by

$$\begin{aligned}\det A = {} & a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} \\ & -a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}.\end{aligned}$$

For a binary matrix B we replace this expression by

$$\begin{aligned}\det B = {} & (b_{11} \cdot b_{22} \cdot b_{33}) \oplus (b_{12} \cdot b_{23} \cdot b_{31}) \oplus (b_{13} \cdot b_{21} \cdot b_{32}) \\ & \oplus (b_{13} \cdot b_{22} \cdot b_{31}) \oplus (b_{11} \cdot b_{23} \cdot b_{32}) \oplus (b_{12} \cdot b_{21} \cdot b_{33})\,.\end{aligned}$$

(i) Calculate the determinant for the binary matrices

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

(ii) Calculate the determinant for the binary matrices

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Problem 14. The finite field $GF(2)$ consists of the elements 0 and 1 (bits) which satisfies the following addition (XOR-operation) and multiplication (AND-operation) tables

$\oplus$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Find the determinant of the *binary matrices*

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

Problem 15. An (n, k) *binary linear block code* is a k-dimensional subspace of the n-dimensional vector space

$$V_n := \{\, (b_0, b_1, \ldots, b_{n-1}) \,:\, \forall\, b_j \;\, b_j \in GF(2) \,\}.$$

Here n is called the length of the code and k the dimension. An (n, k) binary linear block code can be specified by any set of k linear independent codewords $\mathbf{b}_0$, $\mathbf{b}_1$, …, $\mathbf{b}_{k-1}$. One arranges the k codewords into a $k \times n$

binary matrix G. This matrix G is called a *generator matrix* for all the codewords C. Consider the generator matrix ($n = 6$, $k = 3$)

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Find all the codewords.

Problem 16. Consider the generator matrix ((7,4) *Hamming code*)

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Find all the codewords.

Problem 17. A *parity check* for codewords C is an equation of the form

$$(a_0 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus \cdots \oplus (a_{n-1} \cdot b_{n-1}) = 0$$

which has to be satisfied for any $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1}) \in C$. The collection of all vectors $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ forms a vector subspace of V_n. It is denoted by $C^{\perp}$ and is called the *dual code* of C. The dimension of $C^{\perp}$ is $n - k$ and $C^{\perp}$ is an $(n, n-k)$ binary linear block code. Any generator matrix of $C^{\perp}$ is a *parity-check matrix* for C and is denoted by H. Consider the generator matrix ($n = 6$, $k = 3$)

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Find the parity-check matrix H.

Problem 18. A $(7, 4)$ Hamming code can be generated by

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Show that the sixteen codewords are

```
0000000
0001111
0010110
```

```
0011001
0100101
0101010
0110011
0111100
1000011
1001100
1010101
1011010
1100110
1101001
1110000
1111111
```

Show the parity check matrix is

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Problem 19. Study the Lie algebra $s\ell(2, \mathbb{F})$, where $\mathrm{char}\mathbb{F} = 2$.

Chapter 4

Reversible Logic Gates

Reversible logic gates are gates that function in both directions. CMOS implementations of such gates are designed. A special pass transistor logic family is applied: reversible MOS. Many different reversible logic gates are candidates as a universal building blocks. The controlled NOT gate, the Fredkin gate can be implemented. They dissipate very little energy. Owing to their use of reversible truth tables, they are even candidates for zero-power computing. Circuit synthesis take advantage of mathematical group theory. Algorithms have been developed for the synthesis of arbitrary reversible circuits.

Reversible circuits are applicable to nanotechnolgy, quantum and optical computing as well as reducing power in CMOS implementation. In adiabatic circuits, current is restricted to flow across devices with low voltage drop and the energy stored on their capacitors is recycled. One uses reversible energy recovery gate capable to realize functions $\{AND, OR\}$ or $\{NAND, NOR\}$.

Problem 1. For reversible gates the following boolean expression plays an important role

$$(a_{11} \cdot a_{22}) \oplus (a_{12} \cdot a_{21})$$

where $a_{11}, a_{12}, a_{21}, a_{22} \in \{\, 0, 1 \,\}$. It could be considered as the *determinant* of the 2×2 binary matrix

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

Find the inverse of the matrix when it exists.

Problem 2. Find the truth table for the boolean function

$$f(a, a', b, b') = (a \cdot b') \oplus (a' \cdot b)\,.$$

Problem 3. Consider a two input gate (x, y) / two output gate (x', y') given by

$$x' = a \cdot x \oplus b \cdot y \oplus c$$
$$y' = a' \cdot x \oplus b' \cdot y \oplus c'$$

where $a, b, a', b', c, c' \in \{\, 0, 1 \,\}$.
(i) Let $a = 0$, $b = 1$, $a' = 1$, $b' = 0$ and $c = c' = 0$. Find the output (x', y') for all possible inputs (x, y). Is the transformation invertible?
(ii) Let $a = 1, b = 1, a' = 1, b' = 1$ and $c = c' = 0$. Find the output (x', y') for all possible inputs (x, y). Is the transformation invertible?

Problem 4. The *Feynman gate* is a 2 input/2 output gate given by

$$x_1' = x_1$$
$$x_2' = x_1 \oplus x_2$$

(i) Give the truth table for the Feynman gate.
(ii) Show that copying can be implemented using the Feynman gate.
(iii) Show that the complement can be implemented using the Feynman gate.
(iv) Is the Feynman gate invertible?

Problem 5. Consider the *Toffoli gate*

$$T : \{0,1\}^3 \to \{0,1\}^3, \qquad T(a, b, c) := (a, b, (a \cdot b) \oplus c)$$

where $\bar{a}$ is the NOT operation, $+$ is the OR operation, $\cdot$ is the AND operation and $\oplus$ is the XOR operation.

1. Express $NOT(a)$ exclusively in terms of the TOFFOLI gate.
2. Express $AND(a, b)$ exclusively in terms of the TOFFOLI gate.
3. Express $OR(a, b)$ exclusively in terms of the TOFFOLI gate.
4. Show that the TOFFOLI gate is invertible.

Thus the TOFFOLI gate is universal and reversible (invertible).

Problem 6. Consider the *Fredkin gate*

$$F : \{0,1\}^3 \to \{0,1\}^3, \qquad F(a,b,c) := (a, a \cdot b + \bar{a} \cdot c, a \cdot c + \bar{a} \cdot b)$$

where $\bar{a}$ is the NOT operation, $+$ is the OR operation, $\cdot$ is the AND operation and $\oplus$ is the XOR operation.

1. Express $NOT(a)$ exclusively in terms of the FREDKIN gate.
2. Express $AND(a, b)$ exclusively in terms of the FREDKIN gate.
3. Express $OR(a, b)$ exclusively in terms of the FREDKIN gate.
4. Show that the FREDKIN gate is invertible.

Thus the FREDKIN gate is universal and reversible (invertible).

Problem 7. The *Toffoli gate* $\mathrm{T}(x_1, x_2; x_3)$ has 3 inputs (x_1, x_2, x_3) and three outputs (y_1, y_2, y_3) and is given by

$$(x_1, x_2, x_3) \to (x_1, x_2, x_3 \oplus (x_1 \cdot x_2))$$

where $x_1, x_2, x_3 \in \{\, 0, 1 \,\}$, $\oplus$ is the XOR-operation and $\cdot$ the AND-operation. Give the truth table.

Problem 8. A *generalized Toffoli gate* $\mathrm{T}(x_1, x_2, \ldots, x_n; x_{n+1})$ is a gate that maps a boolean pattern $(x_1, x_2, \ldots, x_n, x_{n+1})$ to

$$(x_1, x_2, \ldots, x_n, x_{n+1} \oplus (x_1 \cdot x_2 \cdot \ldots \cdot x_n))$$

where $\oplus$ is the XOR-operation and $\cdot$ the AND-operation. Show that the generalized Toffoli gate includes the NOT-gate, CNOT-gate and the original Toffoli gate.

Problem 9. The *Fredkin gate* $\mathrm{F}(x_1; x_2, x_3)$ has 3 inputs (x_1, x_2, x_3) and three outputs (y_1, y_2, y_3). It maps boolean patterns

$$(x_1, x_2, x_3) \to (x_1, x_3, x_2)$$

if and only if $x_1 = 1$, otherwise it passes the boolean pattern unchanged. Give the truth table.

Problem 10. The *generalized Fredkin gate* $\mathrm{F}(x_1, x_2, \ldots, x_n; x_{n+1}, x_{n+2})$ is a gate is the mapping of the boolean pattern

$$(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}) \to (x_1, x_2, \ldots, x_n, x_{n+2}, x_{n+1})$$

if and only if the boolean product $x_1 \cdot x_2 \cdot \ldots \cdot x_n = 1$ ($\cdot$ is the bitwise AND operation), otherwise the boolean pattern passes unchanged. Let $n = 2$ and $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$. Find the output.

Problem 11. Is the gate $(a, b, c \in \{\, 0, 1 \,\})$

$$(a, b, c) \to (a, a \cdot b \oplus c, \overline{a} \cdot \overline{c} \oplus \overline{b})$$

reversible?

Problem 12. Prove that the *Fredkin gate* is universal. A set of gates is called universal if we can build any logic circuits using these gates assuming bit setting gates are given.

Problem 13. The *half-adder* is given by

$$S = A \oplus B$$
$$C = A \cdot B.$$

Construct a half-adder using two Toffoli gates.

Problem 14. Consider the 3-input/3-output gate given by

$$x_1' = x_1$$
$$x_2' = x_1 \oplus x_2$$
$$x_3' = x_1 \oplus x_2 \oplus x_3.$$

(i) Give the truth table.
(ii) Is the transformation invertible.

Problem 15. Consider the 3-input/3-output gate given by

$$x_1' = x_1$$
$$x_2' = x_1 \oplus x_2$$
$$x_3' = x_3 \oplus (x_1 \cdot x_2).$$

(i) Give the truth table.
(ii) Is the gate invertible?

Problem 16. Consider the 4-input/4-output gate given by

$$\begin{aligned} x_1' &= x_1 \\ x_2' &= x_2 \\ x_3' &= x_3 \\ x_4' &= x_4 \oplus x_1 \oplus x_2 \oplus x_3 \,. \end{aligned}$$

(i) Give the truth table.
(ii) Is the gate invertible?

Problem 17. Show that one Fredkin gate

$$(a, b, c) \to (a,\, \overline{a} \cdot b + a \cdot c,\, \overline{a} \cdot c + a \cdot b)$$

is sufficient to implement the XOR gate. Assume that either $\overline{b}$ or $\overline{c}$ are available.

Problem 18. Show that the map $\mathbf{f} : \{0,1\}^3 \to \{0,1\}^3$

```
abc    xyz
000 -> 000
100 -> 100
010 -> 101
110 -> 011
001 -> 001
101 -> 010
011 -> 110
111 -> 111
```

is invertible. The map describes a reversible half-adder. If $c = 0$, then x is the first digit of the sum $a + b$ and y is the carry bit. If $c = 1$, then z is the first digit of the sum $a + b + c$ and y is the carry bit.

Problem 19. Consider the 3-input/3-output gate given by

$$\begin{aligned} x_1' &= x_1 \oplus x_3 \\ x_2' &= x_1 \oplus x_2 \\ x_3' &= (x_1 \cdot x_2) \oplus (x_1 \cdot x_3) \oplus (x_2 \cdot x_3). \end{aligned}$$

(i) Give the truth table.
(ii) Is the gate invertible?

Problem 20. Consider the 3-input/3-output gate given by

$$x_1' = x_1 \oplus x_3$$
$$x_2' = x_1 \oplus x_2$$
$$x_3' = (x_1 + x_2) \oplus (x_1 + x_3) \oplus (x_2 + x_3).$$

(i) Give the truth table.
(ii) Is the gate invertible?

Problem 21. Consider the 4-input/4-output gate given by

$$x_1' = x_1 \oplus x_3$$
$$x_2' = x_2 \oplus x_3 \oplus (x_1 \cdot x_2) \oplus (x_2 \cdot x_3)$$
$$x_3' = x_1 \oplus x_2 \oplus x_3$$
$$x_4' = x_4 \oplus x_3 \oplus (x_1 \cdot x_2) \oplus (x_2 \cdot x_3)\,.$$

(i) Give the truth table.
(ii) Is the gate invertible?

Problem 22. The NOT, AND and OR gate form a universal set of operations (gates) for boolean algebra. The NAND operation is also universal for boolean algebra. However these sets of operations are not reversible sets of operations. Consider the Toffoli and Fredkin gates

$$TOFFOLI : \{0,1\}^3 \to \{0,1\}^3, \quad TOFFOLI(a,b,c) = (a,b,(a \cdot b) \oplus c)$$

$$FREDKIN : \{0,1\}^3 \to \{0,1\}^3, \quad FREDKIN(a,b,c) = (a, a{\cdot}c{+}\bar{a}{\cdot}b, a{\cdot}b{+}\bar{a}{\cdot}c)$$

where $\bar{a}$ is the NOT operation, $+$ is the OR operation, $\cdot$ is the AND operation and $\oplus$ is the XOR operations.

1. Express NOT(a) exclusively in terms of the TOFFOLI gate.
2. Express NOT(a) exclusively in terms of the FREDKIN gate.
3. Express AND(a,b) exclusively in terms of the TOFFOLI gate.
4. Express AND(a,b) exclusively in terms of the FREDKIN gate.
5. Express OR(a,b) exclusively in terms of the TOFFOLI gate.
6. Express OR(a,b) exclusively in terms of the FREDKIN gate.
7. Show that the TOFFOLI gate is reversible.
8. Show that the FREDKIN gate is reversible.

Thus the TOFFPLI and FREDKIN gates are eachuniversal and reversible (invertible).

Chapter 5

Floating Point Numbers

In C, C++, Java and Perl we have two types of floatimg point number, namely *float* and *double*. For the data type `float` with 32 bits the value is stored as we have

```
sign bit, 8 bit exponent, 23 bit mantissa
```

i.e.

```
  byte 1     byte 2     byte 3    byte 4
SXXX XXXX  XMMM MMMM MMMMM MMMM MMMM MMMM
```

For the data type `double` we have 64 bits in C++. The value of `double` is stored as

```
sign bit, 11 bit exponent, 52 bit mantissa
```

This means

```
  byte 1   byte 2    byte 3    byte 4        byte 8
SXXX XXXX XXXX MMMM MMMM MMMM MMMM MMMM ... MMMM MMMM
```

We can also do bitwise manipulations of floating point numbers, for example on the data type

Problem 1. Write a C++ progran that changes a bit in the floating point number `double` using , is the bitwise OR `|` and the shift operation `<<`.

Problem 2. What is the output of the following C++ program?

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

int main(void)
{
  double x = 3.14159;
  int* p = (int*) &x;
  *(p+1) = *(p+1)^(1<<31); // short cut *(p+1) ^= (1<<31)
  cout << "x = " << x << endl;
  return 0;
}
```

Note that the data type `double` takes 64 bit and the sign bit is at bit position 63. The data type `int` takes 32 bits, `<<` is the shift left operation and `^` is the XOR operation.

Problem 3. C++ and C provide six operators for bit manipulations. These may only be applied to integral operands, that is `char`, `short`, `int` and `long`,whether signed or unsigned.

```
 &  bitwise AND
 |  bitwise OR
 ^  bitwise XOR
 ~  one's complement (unary)
 << left shift
 >> right shift
```

Thus these operators cannot be applied to `double` and `float`. Write a C++ program that can do the bitwise operations AND, OR, XOR, NOT on the data type double.

Problem 4. Write a C++ program using the `bitset` class which converts the memory representation of an arbitrary data type to a bitset of appropiate size and vice versa. In other words store the `sizeof(T)` bytes for an instance of `T` in a bitset. Since `float` and `int` are both 32-bit (on 32 bit architectures) use the program to find an integer with the same binary representation as a given value of type `float`.

Problem 5. What is ouput of the following program

```
// btod.cpp

#include <bitset>
#include <cassert>
#include <iostream>
```

```
using namespace std;

template <class T> class size;

template <> class size<char>
{
 public:
  static const size_t bits = numeric_limits<char>::digits
                           + ((numeric_limits<char>::is_signed) ? 1:0);
  static const size_t chars = 1;
};

template <class T> class size
{
 public:
  static const size_t bits = sizeof(T) * size<char>::bits;
  static const size_t chars = sizeof(T);
};

template <> class size<int>
{
 public:
  static const size_t bits = numeric_limits<int>::digits
                           + ((numeric_limits<int>::is_signed) ? 1:0);
  static const size_t chars = sizeof(int);
};

template <class T, const size_t n>
T map_bitset_to(const bitset<n> &b)
{
 char *cp;
 T d;
 size_t i, j, k = 0;

 assert(n >= size<T>::bits);

 for(i=0, cp=(char*)&d; i < size<T>::chars; i++, cp++)
 {
  *cp = 0;
  for(j=0; j < size<char>::bits; j++, k++)
   if(b.test(k)) *cp |= (1<<j);
 }

 return d;
}

int main(void)
{
```

```
bitset<size<double>::bits>
b(string("0111111111111111111111111111111111111111111111111000000000000000"));
cout << map_bitset_to<double>(b) << endl;
 return 0;
}
```

Problem 6. What is the output of the following program

```
// double.cpp

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

struct double_bits {
 unsigned long mantissa : 52; /* need to modify for 32-bit */
 unsigned int exponent : 11;
 unsigned int sign : 1;
};

union udouble {
 double d;
 double_bits bits;
};

int main(void)
{
 union udouble u;

 cout.precision(numeric_limits<double>::digits);

 u.d = 3.14;
 cout << u.d << endl;
 cout << u.bits.sign << "\t" << u.bits.exponent
      << "\t" << u.bits.mantissa << endl;

 u.bits.sign = 1;
 cout << u.d << endl;
 cout << u.bits.sign << "\t" << u.bits.exponent
      << "\t" << u.bits.mantissa << endl;

 u.bits.exponent++;
 cout << u.d << endl;
 cout << u.bits.sign << "\t" << u.bits.exponent
      << "\t" << u.bits.mantissa << endl;

 u.bits.mantissa--;
```

```
 cout << u.d << endl;
 cout << u.bits.sign << "\t" << u.bits.exponent
      << "\t" << u.bits.mantissa << endl;
 return 0;
}
```

Chapter 6

Cellular Automata

Problem 1. We consider a one-dimensional ring of N sites labelled sequentialy by the index i staring from zero, i.e. $i = 0, 1, \ldots, N-1$. We impose periodic boundary conditions, i.e. $N \equiv 0$. Each site i can take the values $a_i = 0$ or $a_i = 1$. Let a_i evolve as a function $a_i(t)$ of discrete time steps $t = 0, 1, 2, \ldots$ according to the map

$$a_i(t+1) = (a_{i-1}(t) + a_{i+1}(t)) \mod 2.$$

Since the map involves the sum $a_{i-1} + a_{i+1} \bmod 2$, it is a nonlinear map. Give a C++ implementation for this map.

Problem 2. Consider the initial value problem of the two-dimensional cellular automata

$$x_{i,j}(t+1) = (x_{i+1,j}(t) + x_{i,j+1}(t)) \oplus x_{i-1,j}(t) \oplus x_{i,j-1}(t) \oplus x_{i,j}(t), \qquad t = 0, 1, 2, \ldots$$

where i, j are the coordinates of the pixel and $-M \leq i \leq +M$, $-M \leq j \leq +M$ and M is a positive integer. Here $+$ denotes the OR-operation and $\oplus$ the XOR-operation. Periodic boundary conditions are imposed. Write a C++ program that implements this cellular automate. Apply the bitset class of the Standard Template Library.

Problem 3. Consider the initial value problem for the cellular automata

$$x_j(t+1) = x_{j-1}(t) \oplus x_j(t) \oplus x_{j+1}(t), \qquad t = 0, 1, 2, \ldots$$

where $j = \{-15, -14, \ldots, 0, \ldots, 16\}$ and cyclic boundary condition. The initial values at $t = 0$ are $x_0(0) = 1$ otherwise 0. Write a C++ program

that implements this cellular automata using the `bitset` class. Display the results.

Problem 4. Consider the one-dimensional cellular automata with rule 62. Find the map.

Problem 5. (i) Consider the one-dimensional elementary cellular automata

```
111  110 101  100  011  010  001  000
 0    0   0    1    0    1    1    0
```

which is rule 94. Show that starting with a random initial condition the system settles down in a stationary state.
(ii) Consider the one-dimensional elementary cellular automata

```
111  110 101  100  011  010  001  000
 0    1   1    1    1    0    1    0
```

which is rule 50. Show that starting with a random initial condition the system settles down in a state of period 2.

Chapter 7

String Manipulations

Problem 1. Let `s1`, `s2` be two strings. Write a C++ function `squeeze(s1,s2)` that deletes each character in `s1` that matches any character in the string `s2`.

Problem 2. Let `s1` and `s2` be two strings. Write a function `any(s1,s2)`, which returns the first location in the string `s1` where any character from the string `s2` occurs, or `-1` if `s1` contains no characters from string `s2`.

Problem 3. A *substring* of a string **s** consists only of consecutive characters from **s**. A *subsequence* of a string **s** is simply an (ordered) sequence of characters (not necessarily consecutive) from **s**. For example, if **s** = "$ATTGCTA$", then "$AGCA$" and "$ATTA$" are subsequences, but there are not substrings. A common subsequence of two strings is a subsequence of both a them. For example, consider "$ATCTGAT$" and "$TGCATA$". Then "$TCTA$" is a common subsequence of the two strings. The longest common subsequence problem is given two strings **s** and **t** find the longest common subsequence of **s** and **t**.
(i) Write a C++ program that finds the longest common subsequence of two given strings.
(ii) Write a C++ program that finds the longest common substrings of two given strings.

Problem 4. Use the `string` class of C++ and declare an array of strings

```
string* sa = new string[N];
```

Fill this array with strings and concatenate them.

Problem 5. Consider two strings of the same length. Write a C++ program using the `string` class that finds the *Hamming distance*.

Problem 6. Given two strings not necessarily of the same length. Write a C++ program that implements the *Levenshtein distance* (edit distance).

Problem 7. Let $\mathcal{A}$ be a finite alphabet and let $\mathcal{A}^*$ be the set of all finitely long words that can be a written in this alphabet. One denotes by $\mathcal{A}^{\mathbb{N}}$ and $\mathcal{A}^{\mathbb{Z}}$ be the sets of all semi-infinite and infinite sequences of letters from $\mathcal{A}$. Let f be a map from $\mathcal{A} \to \mathcal{A}^*$ that associates with any letter in $\mathcal{A}$ a finite word.

(i) Let $\mathcal{A} = \{a, b\}$ and the map (*Fibonacci sequence*)

$$a \mapsto f(a) = ab, \quad b \mapsto f(b) = a.$$

Give a C++ implementation.
(ii) Let $\mathcal{A} = \{a, b\}$ and the map (*Thue-Morse sequence*)

$$a \mapsto f(a) = ab, \quad b \mapsto f(b) = ba.$$

Give a C++ implementation.
(iii) Let $\mathcal{A} = \{a, b\}$ and the map (*period-doubling sequence*)

$$a \mapsto f(a) = ab, \quad b \mapsto f(b) = aa.$$

Give a C++ implementation.
(iv) Let $\mathcal{A} = \{a, b, c, d\}$ and the map (*Rudin-Shapiro sequence*)

$$a \mapsto f(a) = ac, \quad b \mapsto f(b) = dc, \quad c \mapsto f(c) = ab, \quad d \mapsto f(d) = db.$$

Give a C++ implementation.

Problem 8. Consider the recursion

$$S_{k+1} = S_k^m \cdot S_{k-1}^n, \quad k \geq 2, \quad S_1 = B,\, S_2 = A$$

where $\cdot$ stands for concatenation and n, m are positive integers. One defines

```
n = 1, m = 1    Golden mean or Fibonacci chain
n = 1, m = 2    Silver mean
n = 1, m = 3    Bronze mean
n = 2, m = 1    Copper mean
n = 3, m = 1    Nickel mean
```

Give a C++ implementation utilizing the string class. Count the numbers of A's and B's at each step.

Bibliography

Books

Steeb, W.-H.
Introduction to Assembly Language and C++
International School for Scientfic Computing, 2008

Hardy Y. and Steeb W.-H.
Classical and Quantum Computing with C++ and Java Simulations
Birkhauser-Verlag, Boston (2002)

Steeb W.-H.
Matrix Calculus and Kronecker Product with Applications and C++ Programs
World Scientific Publishing, Singapore (1997)

Mendelson E.
Boolean Algebra and Switching Circuits
Schaum's Outline Series (1970)

Warren H. S.
Hacker's Delight
Addison-Wesly, Boston (2003)

Papers

Index