ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communication
Cours d'Informatique à la section SV
Jobstmann B, Sam J.

ICC

Examen Semestre I

Instructions :

- You have 3 hours to complete this exam (8h00 - 11h).
- The maximum number of points is 100.
- Note that there are also instructions on the back side.
- You must **write using black or dark-blue ink**, do not use pencils or other colors. Do note use **erasable pens** (loss of information due to heat).
- Any paper document is permitted : However, you are not allowed to use a personal computer, nor a cellphone, nor any other electronic device.
- Reply on the answering sheets that were distributed to you **in the dedicated places**. **Do not answer on the statement**.
- Do not attach any additional piece of paper ; **only the distributed answering sheets will be graded**.
- You can reply to the questions in English or in French.
- The exam is made of 7 independent exercises.

  You can start with whatever exercise you want

Exercise	1	2	3	4	5	6	7
Points	12	6	12	15	10	30 (8 + 15 + 7)	15

Exercice 1 : QCM Partie théorie [12 points]

For each multiple choice question you can get two points. For partially correct answers, you will get partial points according to the formula "$2 \cdot \max(0, x_C/C - x_I/I)$", where C and I are the total numbers of correct and incorrect options, respectively, and x_C and x_I are the number of correct and incorrect options that you selected. **The total number of points you receive per question is never negative.**

QCM 1.1 : Complément à deux

Assume we are using an 8-bit two's complement representation of numbers. What is the result of the operation $-96 - 64$ (written in decimal) ?

A) -160 **B)** 32 **C)** 96 **D)** 160

QCM 1.2 : Représentation en virgule flottante

Assume a simplified floating point representation of a positive number using 6 bits as following : 3 bits for the exponent in base 2, 3 bits for the mantissa. We use the two's complement representation for the exponent. Select all correct statements :

A) The decimal number 2.5 can be represented exactly (without rounding errors).

B) The maximum relative error is smaller or equal to $2^{-4} = \frac{1}{16}$.

C) Using this representation we can represent 64 numbers exactly (without rounding errors).

D) The largest number that can be represented exactly (without rounding errors) is 16.

QCM 1.3 : Comparaison asymptotique (O- et Θ notation)

Select all correct statements :

A) $5 \cdot n^3 + n^2 + 100 \in \Theta(n^3)$

B) $2^n \cdot n^2 \in O(n^2)$

C) $10 \cdot n^3 - n^2 \in O(n^2)$

D) $2 \cdot \log_2 n \in \Theta(\log_3 n)$

E) $n^3 + 15 \cdot n^2 + 50 \in \Theta(n^5)$

F) $n \cdot \log n \in \Theta(n^2)$

QCM 1.4 : Mémoire cache

Let's consider the first eight blocks of the main memory (RAM) of a computer. Suppose that each block has two words, i.e., we are interested in the first 16 words in the RAM. Each block is indexed by the address of its first word $0, 2, 4, 6, 8, 10, 12, 14$. In addition, suppose that this computer has a cache that can hold up to 3 blocks, and that the Block 2 and 4 are already loaded into the cache. Given a processor that uses the LRU (Least Recently Used) algorithm and accesses the following sequence of memory addresses :

3 6 4 10 13 5 7 3

How many cache misses does it generate ?

A) 3 **B)** 4 **C)** 5 **D)** 6

QCM 1.5 : Échantillonnage de fréquence pour la reconstruction

Assume that we would like to sample the signal $X(t) = \sin(34\pi t + \pi/2) + 10 \cdot \sin(18\pi t + \pi) + 2 \cdot \sin(12\pi t)$. Which of the following sampling frequencies f_e is the smallest frequency that allows a correct reconstruction of the signal $X(t)$ using the interpolation formula $X_I(t) = \sum_n X(nT_e)\text{sinc}(\frac{t-nT_e}{T_e})$ with $T_e = 1/f_e$.

A) $f_e = 17$ Hz **B)** $f_e = 18$ Hz **C)** $f_e = 34$ Hz **D)** $f_e = 35$ Hz

QCM 1.6 : Effet stroboscopique

Assume that we are sampling the signal $X(t) = 2 \cdot \sin(6\pi t) + 4 \cdot \sin(12\pi t)$ with the sampling frequency $f_e = 10$ Hz and try to reconstruct the signal $X(t)$ using the interpolation formula $X_I(t) = \sum_n X(nT_e)\text{sinc}(\frac{t-nT_e}{T_e})$ with $T_e = 1/f_e$. Which of the following statements is true ?

A) We will observe a stroboscopic effect.

B) The reconstructed signal $X_I(t)$ will have the same frequencies as $X(t)$.

C) The frequency 4 Hz appears in $X_I(t)$.

D) The reconstructed signal $X_I(t)$ will consist of the following frequencies : 2 Hz, 3 Hz, and 6 Hz.

Exercice 2 : Filtre et Spectre [6 points]

Assume that the following signal $X(t)$ is the input signal of an ideal low-pass filter with a cutoff frequency of 10Hz.

$$X(t) = 4\sin(2\pi t) + \sin(9\pi t) + 2 \cdot \sin(12\pi t - \pi/2) + \sin(20\pi t + \pi)$$

Draw (on the answering sheet) the spectrum of the output signal $\hat{X}(t)$ of this ideal low-pass filter.

Exercice 3 : Codage de Huffman's [12 points]

Suppose you would like to send the following sentence (without spaces) to your friend using the minimal number of bits.

EVA CAN I SEE BEES IN A CAVE

In order to create an optimal encoding using Huffman's algorithm and find the minimal number of bits proceed as follows :

3.1 Create a table with the number of appearances per letter (see the table on the answering sheet.)

3.2 Draw a Huffman tree for this table.

3.3 Create a table with a code word for each letter (see the table on the answering sheet.)

3.4 Compute the number of bits needed to send this message with your encoding.

3.5 How many bits do you save with this encoding compare to an ASCII encoding that uses 8 bit per character ?

Exercice 4 : Algorithme et complexité [15 points]

For the following questions recall that we use the notation $L[i]$ to access the i-th element in the list L. We start counting at 1. So, the list L has the elements $L[1], L[2], \ldots$. Furthermore, we assume that comparison, addition, subtraction, multiplication, division, modulo computation, and accessing an element in a list are in $\Theta(1)$.

4.1 : Complexité [4 points]

Consider ALGORITHM2 that takes a list L of student SCIPER numbers of an EPFL course and the size n of the list and prints the number of students in "Corona" group A, B, and C, i.e., the number of students with a SCIPER number that is equal to 0, 1, or 2 modulo 3, respectively. In order to compute the number of students in the different groups, ALGORITHM2 calls ALGORITHM1, which takes a list L of SCIPER number, the size n of the list, and the expected result r of the modulo computation and returns the number of students for which the SCIPER number is equal to r modulo 3. What is the (asymptotic worst-case) complexity of ALGORITHM1 and ALGORITHM2 in terms of n using the Θ-notation ?

Algorithm 1 ALGORITHM1(L, n, r)

```
sum ← 0
for i from 1 to n do
    if L[i] modulo 3 = r then
        sum ← sum +1
return sum
```

Algorithm 2 ALGORITHM2(L, n)

```
groupA ← ALGORITHM1(L, n, 0)
groupB ← ALGORITHM1(L, n, 1)
groupC ← ALGORITHM1(L, n, 2)
print groupA
print groupB
print groupC
```

4.2 : Algorithme [11 points]

In order to compute the entropy of a message m, one has to compute for each letter l that appears in the message the probability p of choosing l uniformly at random from the given message.

Write an algorithm called `compute_letter_probability` that takes as input

1. a message, given as a list of letters, e.g., `message` $= \{A, B, A, B, C\}$,
2. an integer that describes the length of the message, e.g., `n` $= 5$, and
3. a letter, e.g., `letter` $= A$

and returns the probability p. For instance, with the inputs `message` $= \{A, B, A, B, C\}$, `n` $= 5$, and `letter` $= A$, the algorithm should return 0.4. With inputs `message` $= \{A, B, A, B, C\}$, `n` $= 5$, and `letter` $= C$, the algorithm should return 0.2.

What is the (asymptotic worst-case) complexity of your algorithm `compute_letter_probability` in terms of `n` using the Θ-notation ?

Exercice 5 : Programmation[10 points]

A number n is said to be *abundant* if the sum of its proper divisors (i.e., the divisors that do not include the number itself) is larger than n.

1. Write a function C++ `isDivisor` that takes as arguments two numbers and returns `true` if the second divides the first and is different from it (`false` otherwise) ;
2. Write a function C++ `isAbundant` that takes as a parameter a number and returns `true` if it is abundant and `false` otherwise. This function has to use the function `isDivisor` ;
3. Write a function C++ `findAbundant` that takes as a parameter an integer n and returns `true` if at least one abundant number can be found among all of those strictly smaller than n. The number n has the default value of 50. Furthermore, the number of abundant numbers found will also be made available by the function, according to the following example :

```
int k(0);
constexpr int upper(100);
if (findAbundant(k, upper)){ if upper is not specified, its value is 50
   cout << "Il existe " << k << " nombres abondants avant "
        << upper << endl;
}
```

Exercice 6 : Conception de programme et programmation [30 points]

Ice skating schools need your help designing a management tool !

An *ice skating school* is characterised by its name, its address (string of characters), by the *instructors* that work there as well as the *students* who are enrolled.

Each student is characterised by a *name* (a string of characters) and an *age category* (adult or child).

The lessons are given in the morning, afternoon or evening every day of the week except Sunday.

Each instructor is characterised by a name and a weekly schedule (which we suppose is reset at the end of every week). The *schedule* will be given through a table looking like this :

	0 (morning)	1 (afternoon)	2 (evening)
0 (Monday)	*e1*		
1 (Tuesday)			
2 (Wednesday)		*e2*	
3 (Thursday)			
4 (Friday)			*e3*
5 (Saturday)			

Each box of the table must contain information that allows one to know which student is taught by the instructor for a given day and a given time slot. We assume that only one student is taught by an instructor in a given time slot (morning, afternoon or evening). In the example above, the instructor gives a lesson on Monday morning to the student *e1*, on Wednesday afternoon to the student *e2* and on Friday evening to the student *e3*. The rest of the time the instructor does not work.

The following types must be used in your design.

```
// for the days of the week: from 0 (Monday) to 5 (Saturday)
typedef short int Day;
// for the training time slots provided by the instructors:
// 0 (morning), 1 (afternoon), 2 (evening)
typedef short int TimeSlot;
```

Question 1 : structures de données [8 points]

Provide C++ code for the data types/structures allowing to model :

- A student.
- An age category.
- An instructor.
- A weekly schedule.
- An ice skating school.

The definitions of the data structures and types will be given in the order of precedence required by C++. You can of course define other types if necessary and make wise use of the `typedef`.

To model the schedule, think carefully about the type to use for the content of the cells in the table. Indicate in the comments how a cell «without student» is represented.

Question 2 : fonctionnalités [15 points]

Below we give the functions that are necessary for the management of our skating schools. Give the prototypes of these functions. You are not asked to write a complete program nor the body of functions but only prototypes. You can add other functions if you consider it relevant for a good modularization. Nor is it asked to write include directives (`#include<..>`).

1. add an instructor with a given name to a given school (this function will be in charge of creating the instructor) ;
2. enroll a student to a given school (this function will be in charge of creating the student) ;
3. find, without printing it, the most represented age category among the students of a given school ;
4. print the information associated to an instructor (it is necessary to print the name and the schedule ; when the schedule is printed, the name of the students involved will be displayed) ;
5. find, without printing them, all the free time slots of an instructor with a given name and for a given day (this function will allow one, for instance, to know that the instructor "Bob" is available on Monday morning and evening) ;
6. find, without printing it, the number of time slots when a given instructor is busy (count the «non-empty» cells in his/her schedule) ;
7. find, without printing them, all the instructors that have at least one free time slot for a given day ;
8. assign a given student to a given instructor, on a given day and a given time slot ; this function must indicate whether the assignment has been successful or not (this will not be the case if the corresponding cell of the instructor's schedule is already occupied) ;
9. find, without printing it, the most popular instructor (the instructor with the highest number of occupied cells in his/her schedule ; we will suppose that in case there is more than one, only the first one will be returned) ;

Question 3 : programmation [7 points]

1. Give the code implementing function 9.

Use the prototyped functions without writing their bodies. Be sure to maintain a good indentation.

Exercice 7 : Déroulement de programme [15 points]

The following program compiles and runs without errors (C++11).

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  #include <algorithm>
5  using namespace std;
6
7  typedef string Gene;
8  typedef vector<Gene> Genes;
9
10 size_t d(const Gene& g1, const Gene& g2) {
11     size_t count(0);
12     int t1(g1.size());
13     int t2(g2.size());
14     for (size_t i(0); i < std::min(t1,t2); ++i) {
15         if (g1[i] != g2[i]) ++count;
16     }
17     return count + abs(t1-t2);
18 }
19
20 void o(const Genes& gs, const Gene& r) {
21     for (const auto& g : gs) {
22         cout << d(g,r) << endl;
23     }
24 }
25
26 bool f(const Gene& g) {
27     const char S('c');
28     if (g.size() == 0) return false;
29     if (g.size() == 1) {
30         return g[0] == S;
31     }
32
33     size_t l(0);
34     size_t h(g.size());
35     size_t m((l+h)/ 2);
36     if (g[m] == S) return true;
37     // substr extrait la sous-chaîne de longueur m depuis l'indice l
       compris:
38     if (g[m] > S) return f(g.substr(l, m));
39     else return f(g.substr(m, h-m));
40 }
41 // SUITE SUR LA PAGE SUIVANTE -->
42
43
44
45
46
```

```
47
48
49 void o (const Genes& gs) {
50     Genes clone(gs);
51     for (size_t i(0); i < clone.size(); ++i) {
52         auto g(clone[i]);
53   // trie g par ordre alphabétique:
54         sort(g.begin(), g.end());
55         if (f(g)) {
56             cout << gs[i] << endl;
57         }
58     }
59 }
60
61 int main() {
62     Gene ref("cgtatttaaa");
63
64     Genes seqs({ "tgtatttaaa",
65                  "tttgtttaaa",
66                  "tttatttaaa",
67                  "cgtatttggga",
68                  "uu"
69                });
70     cout << "xxxxx" << endl;
71     cout << ref << endl;
72     cout << "-----" << endl;
73     o(seqs,ref);
74     cout << "/////" << endl;
75     o(seqs);
76     return 0;
77 }
```

1. What does the function `f` do conceptually ? Which known algorithm is used here ?
2. What is the time complexity of the function `f` in terms of the size `n` of the sequence `g` ?
3. What does the program print ? Briefly explain how it works. You are not meant to paraphrase the code, but to *explain* the steps and the program flow.
4. Assume for simplicity that the set `gs` contains `n` lines and each line is a string of length `n`. What is the time complexity in terms of `n` of the function `o` in line 49. Assume that the complexity of the `sort` algorithm is $\Theta(n \log(n))$. Justify briefly.