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
B. Jobstmann, J. Sam

ICC

Examen Semestre I

Instructions :

- You have 3 hours to complete this exam (8h00 - 11h).
- The maximum number of points is 100.
- **Note that instructions are also on the backside.**
- You must **write using black or dark-blue ink**, do not use pencils or other colors.
- You are allowed to bring 1-3 reference books and four double-sided sheets of paper notes for each part of the course (four for theory and four for practice). A printed and stapled version of the BOOC counts as a reference book. Any electronic material (including a calculator) is forbidden.
- 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 5 independent exercises (2 for theory and 3 for programming).

You can start with whatever exercise you want

Exercise	1	2	3	4	5
Points	16	20	15	34 (12 + 22)	15

continue on the back ☞

Exercise 1 : MC Theory part [16 points]

For each multiple-choice question, you can get p points. For partially correct answers, you will get partial points according to the following procedure :

$+p/N$ points if you select a correct answer and N is the total number of correct options,

0 points if you do not answer,

$-p/M$ points if you select an incorrect answer and M is the total number of incorrect options.

QCM 1.1 : Landau Notations [1 point]

Consider the following statements that use different Landau notations. Which statements are correct ? (**Multiple correct answers are possible.**)

A) $2^n + n^2 \in O(n^2)$

B) $n \cdot \log n + 4 \cdot n^3 + n \in \Omega(n \cdot \log n)$

C) $5 \cdot n \cdot \log n + 4 \cdot n^2 + n \in \Theta(n^3)$

D) $\log_{10}(n^2) \in \Theta(\log_2(n))$

Algorithms and their Complexity

Consider the following algorithms ALGO1 and ALGO2. ALGO1 takes a list L of numbers as input and returns a number. The second algorithm ALGO2 also takes a list L of numbers as input and returns a list of numbers. 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 $L[1], L[2], \ldots L[n]$, where n is the number of elements in the list. In addition, the algorithm SIZE(L) takes the list L as an input and returns the number of elements in L.

Algorithm 1 ALGO1(L)

```
1 : $n \leftarrow$ SIZE$(L)$
2 : $r \leftarrow 1$
3 : for $i$ from 1 to $n$ do
4 :     if $L[i] > L[r]$ then
5 :         $r \leftarrow i$
6 : return $r$
```

Algorithm 2 ALGO2(L)

```
1 : $n \leftarrow$ SIZE$(L)$
2 : $R \leftarrow L$
3 : $i \leftarrow 1$
4 : while $i < n$ do
5 :     $p \leftarrow$ ALGO1$(R)$
6 :     $R[p] \leftarrow 0$
7 :     $i \leftarrow 2 \cdot i$
8 : return $R$
```

QCM 1.2 : Execution of Algorithm 1 [1 point]

Which of the following numbers is returned when ALGO1 is called with the list $\{1, 6, 8, 9, 4, 6, 2, 4, 9\}$?

A) 2
B) 4
C) 6
D) 9

QCM 1.3 : Execution of Algorithm 2 [1 point]

Which of the following lists is returned when ALGO2 is called with the list $\{1, 6, 8, 9, 4, 6, 2, 4, 9\}$?

A) $\{1, 6, 8, 0, 4, 6, 2, 4, 9\}$
B) $\{1, 6, 8, 0, 4, 6, 2, 4, 0\}$
C) $\{1, 6, 0, 0, 4, 6, 2, 4, 0\}$
D) $\{1, 0, 0, 0, 4, 6, 2, 4, 0\}$

QCM 1.4 : Complexity [1 point]

Let n be the length of the input list L, i.e., $n = \text{size}(L)$, and assume that $T(n)$ describes the number of instructions that ALGO2(L) executes in the worst case depending on n. We assume that the number of instructions to compute the size of a list (SIZE) is constant. Which statement is correct ?

A) $T(n) \in \Theta(\log n)$
B) $T(n) \in \Theta(n)$
C) $T(n) \in \Theta(n \cdot \log n)$
D) $T(n) \in \Theta(n^2)$

QCM 1.5 : Countability [2 points]

Which of the following statements are correct ? (**Multiple correct answers are possible.**)

A) The set of all functions $f : \{0, 1\} \to \mathbb{N}$ is countable.
B) The set of all infinite sequences of numbers $\{0 - 9\}$ is countable.
C) The set of integers non-divisible by 5 $\{z \in \mathbb{Z} \mid z \mod 5 \neq 0\}$ is countable.
D) The set of all programs written in C++ is **not** countable.

Recursive Algorithms and their Complexity

Consider the algorithm ALGO3 below, which takes a list of numbers L as input and returns a list of numbers. We use the notations PUSH(L, x) to add the element x to the end of the list L and POP(L) to remove the last element in the list L. In addition, SIZE(L) refers to the size of the list L.

Algorithm 3 ALGO3(L)

```
1 : $n \leftarrow$ SIZE$(L)$
2 : if $n \leq 1$ then
3 :     return $L$
4 : else
5 :     $l \leftarrow L[n]$
6 :     POP$(L)$
7 :     $M \leftarrow$ ALGO3$(L)$
8 :     PUSH$(M, 0)$
9 :     for $i$ from $n$ to 2 going down do
10 :        $M[i] \leftarrow M[i-1]$
11 :     $M[1] = l$
12 : return $M$
```

QCM 1.6 : Recursive Algorithm [2 points]

Which of the following lists is returned when **Algorithm 3** is called with the input $L = \{3, 4, 1, 7, 9, 2\}$, i.e., ALGO3$(\{3, 4, 1, 7, 9, 2\})$.

A) $\{2, 9, 7, 1, 4, 3\}$
B) $\{3, 4, 1, 7, 9, 0\}$
C) $\{0, 0, 0, 0, 0, 0\}$
D) $\{1, 2, 3, 4, 7, 9\}$

QCM 1.7 : Complexity [2 points]

What is the (asymptotic worst-case) complexity of the algorithm ALGO3 if we assume that adding an element to the end of the list or removing an element from the end of the list is in $\Theta(1)$ and n is the size of the input list L ?

A) $\Theta(\log n)$
B) $\Theta(n)$
C) $\Theta(n \cdot \log n)$
D) $\Theta(n^2)$

QCM 1.8 : Circuits [2 points]

Recall that we use the symbols shown in Figure 1 to represent AND, OR, and NOT gates, respectively. Note that two crossing wires are only connected if there is a small filled black circle at the crossing.

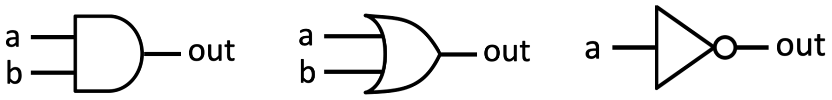

FIGURE 1 – Symbols for an AND gate (left), an OR gate (middle), and a NOT gate (right).

Consider the truth table in Table QCM 1.8 : . Select **all** circuits given in Figure 2 that compute the function **f**.

TABLE 1 – truth table of function f

a	**b**	**f**
0	0	1
0	1	1
1	0	0
1	1	1

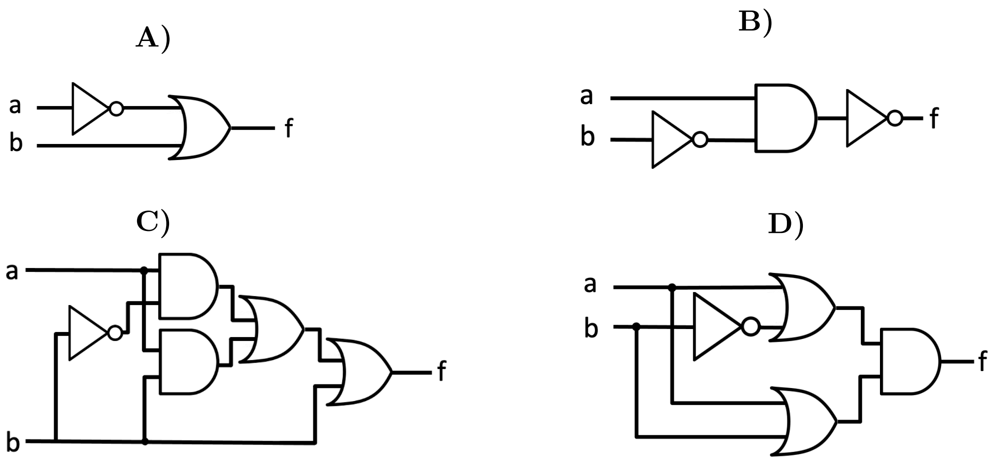

FIGURE 2 – Circuits

QCM 1.9 : Assembly Language [2 points]

Assume you are given a CPU with ten registers (`r0` to `r9`) and the basic instructions given in Table 2.

TABLE 2 – Instruction Set

Instruction	Meaning
`copy r0, c`	$r0 \leftarrow c$: copy a constant c into register r0
`copy r0, r1`	$r0 \leftarrow r1$: copy the value of register r1 to register r0
`add r0, r1, c`	$r0 \leftarrow r1 + c$: add constant c to the register r1 and put the result in r0
`add r0, r1, r2`	$r0 \leftarrow r1 + r2$: add the values of registers r1 and r2 and put the result in r0
`sub r0, r1, c`	$r0 \leftarrow r1 - c$: subtract the constant c from the register r1 and put the result in r0
`sub r0, r1, r2`	$r0 \leftarrow r1 - r2$: subtract the value of register r2 from r1 and put the result in r0
`mul r0, r1, r2`	$r0 \leftarrow r1 \cdot r2$: multiply the values of register r1 and r2 and put the result in r0
`div r0, r1, r2`	$r0 \leftarrow r1/r2$: integer division of register r1 by r2 ; the result is stored in r0
`jump n`	Jump to line n of the program
`jump_gt r0, r1, n`	Jump to line n if the value in r0 is greater than the value in r1 ($r0 > r1$)
`jump_ne r0, r1, n`	Jump to line n if the value in r0 is not equal to the value in r1 ($r0 \neq r1$)
`stop`	Stop the program

Consider the assembly program shown on the right and the three programs below, each takening two integers x and y as input and returning an integer. The slash operator (/) in the programs refer to an integer division, e.g., $5/2 = 2$. Which of the programs (A, B, or C) computes the same result as the assembly program, if x is stored in `r0`, y is stored in `r1`, and r is stored in `r2`.

```
 1: copy r2, 1
 2: copy r3, 1
 3: jump_gt r3, r1, 12
 4: copy r4, 1
 5: div r5, r3, 2
 6: mul r5, r5, 2
 7: jump_ne r5, r3, 9
 8: copy r4, r0
 9: mul r2, r2, r4
10: add r3, r3, 1
11: jump 3
12: stop
```

A)

```
A($x, y$)
1 : $r \leftarrow 1$
2 : if $y > 0$ then
3 :     for $i$ from 1 to $y$ do
4 :         $z \leftarrow 1$
5 :         if $(i/2) \cdot 2 = i$ then
6 :             $z \leftarrow x$
7 :         $r \leftarrow r \cdot z$
8 : return $r$
```

B)

```
B($x, y$)
1 : $r \leftarrow 1$
2 :
3 : for $i$ from 1 to $y$ do
4 :     $z \leftarrow 1$
5 :     if $(i/2) \cdot 2 \neq i$ then
6 :         $z \leftarrow x$
7 :     $r \leftarrow r \cdot z$
8 : return $r$
```

C)

```
C($x, y$)
1 : $r \leftarrow 1$
2 :
3 : for $i$ from 1 to $y$ do
4 :     $z \leftarrow 1$
5 :     if $(i/2) = i$ then
6 :         $z \leftarrow x$
7 :     $r \leftarrow r \cdot z$
8 : return $r$
```

QCM 1.10 : Memory [2 points]

Let's consider the first eight blocks of the main memory (RAM) of a computer. Suppose that each block has 16 words, i.e., we are interested in the first 128 words in the RAM. Each block is indexed by the address of its first word $0, 16, 32, 48, 64, 80, 96, 112$. In addition, suppose that this computer has a cache that can hold up to 3 blocks, and that the Block 16 and 96 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 :

$$19 \quad 8 \quad 30 \quad 99 \quad 77 \quad 32 \quad 31 \quad 79$$

How many cache misses does it generate ?

A) 2
B) 3
C) 4
D) 5

TABLE 3 – Part of the ASCII table

Letter	ASCII	Letter	ASCII	Letter	ASCII	Letter	ASCII
A	65	N	78	a	97	n	110
B	66	O	79	b	98	o	111
C	67	P	80	c	99	p	112
D	68	Q	81	d	100	q	113
E	69	R	82	e	101	r	114
F	70	S	83	f	102	s	115
G	71	T	84	g	103	t	116
H	72	U	85	h	104	u	117
I	73	V	86	i	105	v	118
J	74	W	87	j	106	w	119
K	75	X	88	k	107	x	120
L	76	Y	89	l	108	y	121
M	77	Z	90	m	109	z	122

Exercise 2 : Writing an algorithm [20 points]

1. Assume you are given two algorithms called letterToInt and intToLetter that convert a letter into its integer representation and back following the ASCII encoding that is given in Table 3, e.g., letterToInt(A) = 65 and intToLetter(100) = d. Use these algorithms to write an algorithm called toLowerCase(L) that takes a word that is given as a list of letters, e.g., {L,a,u,s,a,n,n,e}, returns another list in which all upper-case letters (A-Z) are replaced by lower-case letters (a-z), e.g., toLowerCase({L,a,u,s,a,n,n,e}) should return {l,a,u,s,a,n,n,e}. (Note that given the ASCII value of an uppercase letter, it is sufficient to add a fixed value to obtain the ASCII value of the corresponding lowercase letter.)
2. Write an algorithm called sortLetters(L) that given a list of letters returns an alphabetically sorted version of this list. All upper-case letters needs to be put before lower-case letters (like in the ASCII table), e.g., sortLetters({L,a,u,s,a,n,n,e}) should return {L,a,a,e,n,n,s,u}. Recall that you can use the algorithm sort that we saw in the lecture to get a sorted version of a list of integers.
3. Write an algorithm called isAnagram(L_1, L_2) that takes two words as inputs and returns true if the first word is an anagram of the second one, otherwise it returns false. The word is an anagram of another word if it can be build by rearranging the letters of the other word. We make no difference between lower and upper-case letters, e.g., “silent” is an anagram of “LISTEN”. Each word is given as a list of letters, e.g., {s,i,l,e,n,t} and {L,I,S,T,E,N}. You can use the algorithms toLowerCase and sortLetters from the previous tasks.
4. What is the complexity of isAnagram(L_1, L_2) in terms of the m, where m is the maximum number of letters in the first and the second word, i.e., $m = \max(n_1, n_2)$, where n_i is the number of elements in L_i. (Recall that you can assume that the number of instructions executed in size(L) is in $\Theta(1)$ and that the number of instruction executed in sort(L) is in $\Theta(n \cdot \log n)$, where n is the number of elements in L.)

Exercise 3 : C++ Syntax and basics [15 points]

1. For each of the following instructions (taken separately) indicate whether it corresponds to correct or incorrect code in C++. If the code is correct, describe what the instructions mean.

1. `void g(double a=4.5);`
2. `void g(double a=4.5, int b);`
3. `void g(double, int);`
4. `void g(double a==4.5);`
5. `double (g*)(double);`
6. `f(int a) = 5;`
7. `f(int a) = (int a);`
8. `a = f(b);`
9. `f(f(a));`
10. `f(*g);`
11. `f(g&);`
12. `void f(const int& g);`

2. Consider the following statement :

```
struct Image { };
```

Which of the following statements are correct ?

1. `Image` is a type.
2. `Image` is a variable.
3. `Image` is an array.
4. `Image a;` is a correct declaration.
5. `Image a; a.pixel=2;` is correct code.
6. `typedef Ptr Image*;` is a valid instruction.
7. `typedef Image* Ptr;` is a valid instruction.

Exercise 4 : Program design and programming [34 points]

You are helping *pet sitting companies* organize their weekly workload. Each company employs a group of pet sitters. Each *pet sitter* is characterized by his or her *name* and *timetable*. A timetable indicates for each day of the week the assigned shifts. (We assume a maximum of one shift per day.) A *shift* is characterized by the *address* and the *type* of the animal, and whether or not it has to be taken for a *walk*. In addition to its set of pet sitters, a company is characterized by its *name*, the *base rate* applied to a shift (a *double*) and the extra charge (expressed as percentage of the base rate) applied in the case of a walk (a *double*).

The following types are provided and must be used :

```
enum Day {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};
enum Animal {DOG, CAT, FISH, NONE}; // types  d'animaux
```

Question 1 : Data structures/types [12 points]

Give C++ code that can be used to model the data types/structures : a shift, a timetable, a pet sitter, and a pet sitting company.

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

Question 2 : Functions [22 points]

The following functionalities are required for our pet sitting companies. Provide the prototypes for these functions. You are not asked to write a complete program or the function bodies, but only the prototypes. It is also not required to write inclusion directives (`#include<..>`).

You should add other functions if you find them relevant for **good modularization**. You can also **indicate in comments if a function uses others** among those requested if it is necessary for understanding. Please also consider the edge cases.

1. Display all the data for a given company, including all its pet sitters. For each sitter the name and the timetable are displayed. Displaying a timeable means displaying for each day of the week, the address of the animal being looked after, its type and whether a walk is necessary.
2. Add a pet sitter to a given company. The sitter will be created in the function from its characteristic data and with an empty timetable, i.e., there are no shifts assigned for any of the week days.
3. Add a shift to a given sitter for a given day. This shift will be created in the function from its characteristic data. If the day in question is already occupied by a shift, the latter will be replaced by the new shift.
4. For a given company, find, without displaying it, a pet sitter with the highest number of shifts in his or her timetable and a sitter with an empty schedule.
5. For a given company, find, without displaying, the day of the week with the most shifts.
6. For a given company, find the type of animal with the greatest number of shifts.
7. For a given company, find the names of available pet sitters for a given day.
8. For a given company, calculate the weekly income for a given pet sitter (sum of the rates for all his/her shifts, including the extra charge for walks).

Exercise 5 : Program flow [15 points]

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

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

typedef string Data;

struct N {
  Data data;
  N* left;
  N* right;
};

string  p(string prefix, int rep){
  if (rep==0) return "";
  return (prefix + p(prefix, rep-1));
}

void p(N* root, int level, int depth=3)
{
  if (root == nullptr) {
    return;
  }
  p(root->left, level + 1, depth);
  p(root->right, level + 1, depth);

  if (depth == level) {
    cout << root->data << " ";
  }
}

N* init(Data data)
{
  N* result(new N());
  result->data = data;
  result->left = nullptr;
  result->right = nullptr;
  return result;
}

N* c(N* n, Data left, Data right, int height){
  if (n == nullptr || height ==0) return n;
  n->data=left+right;
  n->left = init(left);
  n->right = init(right);
  c(n->left, left, right,  height-1);
  c(n->right, left, right, height-1);
  return n;
}
```

```

int main()
{
  const Data LEFT("AC");
  const Data RIGHT("GT");
  constexpr int  K(3);
  N* root(init(""));
  root = c(root, LEFT, RIGHT, K);
  for (int i(1); i<=K; ++i){
    cout << p(" ", K-i+1);
    p(root, 1, i);
    cout << endl;
  }
  return 0;
}
```

1. What do the functions `p` do conceptually ?
2. What does the function `c` do conceptually ?
3. What is the complexity of the function `c` in terms of the input `height` ? Justify briefly. (Note that changing the other parameters has no impact on the number of instructions executed in the worst case.)
4. What does the program display ? Briefly *explain* how it works. The aim here is not to paraphrase the code, but to explain the program's steps.