ICC SSV Theory Mid-Term Exam

ATTENTION: solutions are visible in the document

Instructions

- This exam has descriptions in two languages: English in black and French in blue. Figures, tables, and code snippets are only shown once in black. If you need a translation, please ask one of the assistants.
- You will have 1h30 to complete this exam.
- Prepare something to write and erase and your Camipro card. You can use the last two empty pages of the exam as scratch paper. If you need more scratch paper, ask an assistant.
- **No documents, computers, tablets, phones, calculators etc. are allowed.**
- Separate Page 3 from the rest of the document.
- Write your SCIPER number, name, and first name on the top of Page 3.
- Write your answers in the table on Page 3 and Page 4 in clearly readable letters.
- At the end of the exam, hand in Page 3 and 4. Only the answers written on these pages will be taken into account.
- All but the last two questions are multiple choice questions. For each multiple choice question you can get at most one point. For partially correct answers, you will get partial points according to the formula "$\max(0, x_C/C - x_I/I)$", 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. So, if the question was two correct and two incorrect options, and you select one option correctly, you will get 0.5 points. If you select a correct and an incorrect option you will get 0 points. If you select only incorrect options you will also get 0 points. The total number of points per question is never negative.

Les instructions en français sont au verso.

Instructions

- Cet examen se décline en deux langues: en anglais en noir et en français en bleu. Les figures, les tableaux et les extraits de code ne sont affichés qu'une seule fois en noir. Si vous avez besoin de traduction, veuillez demander à l'un des assistants.
- Vous disposez de 1h30 pour faire cet examen.
- Préparez de quoi écrire, de quoi effacer et votre carte Camipro. Vous pouvez utiliser les deux dernières pages comme papier brouillon. Si vous avez besoin de plus de papier brouillon, demandez à un assistant.
- **Aucun document, ordinateur, tablette, téléphone, calculatrice etc. n'est autorisé.**
- Détachez la page 3 de votre copie.
- Indiquez vos numéro SCIPER, nom et prénom en haut de la page 3.
- Indiquez vos réponses dans le tableau de la page 3 et dans le tableau de la page 4 en lettres clairement lisibles.
- En fin d'examen, rendez uniquement les pages 3 et 4. Seules les réponses sur ces pages seront prises en compte.
- Toutes les questions, sauf les deux dernières, sont à choix multiple. Pour chaque question à choix multiple vous pouvez obtenir au plus un point. Pour des réponses partiellement correctes, vous recevrez une fraction des points selon la formule "$\max(0, x_C/C - x_I/I)$", où C est le nombre de d'options correctes et I le nombre d'options incorrectes et x_C et x_I sont le nombre d'options correctes ou incorrectes que vous avez choisies respectivement. Donc par exemple, si les propositions contenaient 2 propositions correctes et 2 propositions incorrectes et vous avez sélectionné une seule option correctement, vous recevrez 0.5 points. Si vous sélectionnez une option correcte et une option incorrecte vous recevrez 0 point. Si vous ne sélectionnez que des options incorrectes, vous recevrez 0 point également. **Vous ne recevrez jamais de point négatifs pour une question**.

SCIPER	
Family name Nom de famille	
First name Prénom	

Question	Answer	Points	Max. Points
1			1
2			1
3			1
4			1
5			1
6			1
7			1
8			1
9			1
10			1
11			1
12			1
13			1
14	Please answer below.		2
15	Please answer on the next page.		5
Total			20

Answer of Question 14	**Points:**	/ 2

Answer of Question 15	**Points:**	/	5

(a)

(b)

(c)

1 Question: Binary System

How many distinct pieces of information can be represented with 7 bits?
Combien d'éléments d'information distincts peut on représenter en utilisant 7 bits ?

A. 63

B. 64

C. 127

D. 128

D. $2^7 = 128$.

2 Question: Positive Integer Representation

Which of the following binary patterns represents the number 33 (in decimal), if the binary patterns represent non-negative natural numbers.
Lequel des motifs binaires suivants représente le nombre 33 (en décimal), en admettant que les motifs binaires représentent des nombres naturels non-négatifs.

A. 010001

B. 100001

C. 011001

D. 010011

B. $1 \cdot 2^5 + 1 \cdot 2^0 = 33$.

3 Question: Binary Division

Consider the binary pattern 001100 that represents a non-negative natural number. What is the result of dividing this number by 4 (in decimal)?
Considérez le motif binaire 001100 qui représente un nombre naturel non-négatif. Quel est le résultat de la division de ce nombre par 4 (en décimal) ?

A. 3 (in decimal)

B. 4 (in decimal)

C. 000110 (binary pattern)

D. 011000 (binary pattern)

A. $001100 \gg 2 = 000011 = 3$ (in decimal)

4 Question: Two's Complement

Assume we are using an 8-bit two's complement representation of numbers. What is the result of the operation $65 + 65$ (written in decimal)?
Supposons que soit utilisée une représentation des nombres en complément à deux sur 8 bits. Quel est le résultat de l'opération $65 + 65$ (écrit en décimal) ?

A. 130

B. -130

C. -126

D. -2

C. -126
$65 = 01\_00\_00\_01$
$01\_00\_00\_01 + 01\_00\_00\_01 = 10\_00\_00\_10$
$10\_00\_00\_10 = -(\text{2's complement of } 10\_00\_00\_10) = -01\_11\_11\_10 = -126$

Or, alternatively in decimal: $65 + 65 = 130$. Since $130 > 127$, we get $-(256 - 130) = -126$

5 Question: Floating Point Representation

Assume a simplified floating point representation of a positive number using 3 bits as following: 1 bit for the exponent in base 2, 2 bits for the mantissa. Which of the following statements is correct?

A. The decimal number 3.75 can be represented exactly (without rounding errors).

B. In this representation, only 8 different values in total can be represented exactly (without rounding errors).

C. The maximum relative error is $2^{-3} = 0.125$.

D. The decimal number 2.6 can be represented exactly (without rounding errors).

E. The distance between two numbers that can be represented exactly is always 0.25.

Soit une représentation simplifiée en virgule flottante d'un nombre positif sur 3 bits, faite de la manière suivante: 1 bit pour l'exposant en base 2, 2 bits pour la mantisse. Laquelle des affirmations suivantes est vraie ?

A. Le nombre décimal 3.75 peut être représenté exactement (sans erreurs d'arrondi).

B. Avec cette représentation, seuls 8 valeurs différentes au total peuvent être représentées exactement (sans erreurs d'arrondi).

C. L'erreur relative maximum est $2^{-3} = 0.125$.

D. Le nombre 2.6 (en base 10) peut être représenté exactement (sans erreurs d'arrondi).

E. La distance entre deux nombres représentés exactement est toujours 0.25.

B
1 correct: 1 point
4 incorrect: -0.25 points

6 Question: Recursion

Algorithm 1 ALGORITHM1(n)

```
if n = 1 then
    return 1
else
    return ALGORITHM1(n − 1)+n · n · n
```

Algorithm 2 ALGORITHM2(n)

```
s ← 1
for i from 2 to n do
    s ← s + i · i · i
return s
```

Consider the two algorithms ALGORITHM1 and ALGORITHM2 above. The input is a positive integer $n \geq 1$. **Select all statements that are true.**

A. ALGORITHM1 is recursive and terminates when $n = 1$.

B. ALGORITHM1 calculates the cube of the input n (i.e., n^3).

C. ALGORITHM1 calculates the sum of the first n cubes (i.e., $\sum_{i=1}^{n} i^3$).

D. ALGORITHM1 and ALGORITHM2 solve different problems.

Considérez les deux algorithmes ALGORITHM1 et ALGORITHM2 ci-dessus. L'entrée est un nombre entier $n \geq 1$. **Sélectionnez toutes les affirmations qui sont vraies.**

A. ALGORITHM1 est récursif et se termine quand $n = 1$.

B. ALGORITHM1 calcule le cube de l'entrée n (i.e., n^3).

C. ALGORITHM1 calcule la somme des n premiers cubes (i.e., $\sum_{i=1}^{n} i^3$).

D. ALGORITHM1 et ALGORITHM2 résolvent des problèmes différents.

A and C

7 Question: Big O Notation

Select **all** statements that are true.
Choisissez **toutes** les affirmations vraies.

A. $5 \cdot n^2 + 5000$ is in $O(n^2)$

B. $5 \cdot n^2 + 5000$ is in $O(n^3)$

C. $5 \cdot n^3 + 5000$ is in $O(n^2)$

D. $5 \cdot \log_2(2 \cdot n)$ is in $O(\log_2(n))$

E. $\log_5(n)$ is in $O(\log_2(n))$

F. 2^n is in $O(n^2)$

A, B, D, E
4 correct (0.25 points for selecting a correct option)
2 incorrect (-0.5 points for an incorrect option)

Algorithm and Complexity

Consider the following algorithm that takes an unsorted non-empty list L with n elements as input and returns another list M with the same or less than n elements. 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 $L[1], L[2], \ldots L[n]$. We use $M.append(k)$ to denote that element k is added to the end of the list M.
Considérez l'algorithme suivant qui prend une liste non-triée et non-vide L de n éléments en entrée, et qui retourne une autre liste M avec le même nombre n d'éléments ou moins. Nous utilisons la notation $L[i]$ pour accéder au i-ème élément de la liste L. Nous commençons à compter à partir de 1. La liste L contient donc les éléments $L[1], L[2], \ldots L[n]$. Nous utilisons $M.append(k)$ pour indiquer que l'élément k est ajouté à la fin de la liste M.

Algorithm 3 ALGORITHM3(L, n)

```
1:  k ← 1
2:  M ← empty list
3:  M.append(L[k])
4:  for i from 2 to n do
5:      j ← i − 1
6:      while L[i] ≠ L[j] and j > 0 do
7:          j ← j − 1
8:      if j = 0 then
9:          M.append(L[i])
10: return M
```

8 Question: Algorithm

What does ALGORITHM3 return for the input $\{-6, 4, 1, -5, 4, 7\}$?
Que retourne ALGORITHM3 avec l'entrée $\{-6, 4, 1, -5, 4, 7\}$?

A. $\{-6, -5, 1, 4, 4, 7\}$

B. $\{-6, 4, 1, -5, 4, 7\}$

C. $\{-6, 4, 1, -5, 7\}$

D. $\{-6, 1, -5, 4, 7\}$

C. The algorithm returns a list without duplicates. It keeps the first element and removes all subsequent duplicates.

9 Question: Complexity

What is the (asymptotic worst-case) complexity of ALGORITHM3, if we assume that the function call to $M.append$ is in $O(1)$?

A. $O(1)$

B. $O(\log(n))$ but not $O(1)$

C. $O(n)$ but not $O(\log(n))$

D. $O(n^2)$ but not $O(n)$

Quelle est la complexité (asymptotique dans le pire des cas) de ALGORITHM3 si nous supposons que l'appel de fonction $M.append$ est en $O(1)$?

A. $O(1)$

B. $O(\log(n))$, mais pas $O(1)$

C. $O(n)$, mais pas $O(\log(n))$

D. $O(n^2)$, mais pas $O(n)$

D. In Iteration i of the outer loop, the inner loop has at most i iterations (from $i-1$ to 0), therefore $T(n) = \sum_{i=2}^{n} i = (n \cdot (n+1)/2) - 1 = O(n^2)$

10 Question: Correction

If an algorithm tries to access an element at an index that does not exist, e.g., if L has 5 elements and the algorithm tries to access $L[6]$, then it will generate an error (known as index-out-of-bound error). ALGORITHM3 will generate such an error, if the given list L is empty. Which of the following modifications will remove this error for an empty input list? Select a modification that repairs the program.

A. Replace Line 3 with $M.append(L[0])$

B. Add the following statement between Line 2 and 3:
if $n = 0$ **then return** empty list

C. Replace Line 1 with $k \leftarrow 0$

D. Add the following statement after Line 3:
if $n = 0$ **then return** empty list

Si un algorithme essaie d'accéder à un élément se trouvant à un index qui n'existe pas, e.g., si L a 5 éléments et l'algorithme essaie d'accéder à $L[6]$, alors l'exécution résultera en une erreur (désignée sous le nom d'erreur de type index-out-of-bound). ALGORITHM3 génère une telle erreur si la liste d'entrée L est vide. Laquelle des modifications suivantes supprimera cette erreur pour une liste d'entrée vide ? Choisissez une modification qui réparera le programme.

A. Remplacer la ligne 3 par $M.append(L[0])$

B. Ajouter la ligne suivante entre les lignes 2 et 3:
if $n = 0$ **then return** empty list

C. Replacer la ligne 1 par $k \leftarrow 0$

D. Ajouter la ligne suivante après la ligne 3:
if $n = 0$ **then return** empty list

B

11 Question: Countability

Which of the following statements are correct? **Select all of them.**

A. The set of integers is countable.

B. The set of English sentences is countable.

C. The set of triples (a, b, c) of integers is countable.

D. The set of functions from $\mathbb{N} \to \mathbb{N}$ is countable.

E. The set of rational numbers is countable.

Lesquelles des assertions suivantes sont vraies ? **Sélectionnez-les tous.**

A. L'ensemble des entiers est dénombrable.

B. L'ensemble des phrases en anglais est dénombrable.

C. L'ensemble des triplés d'entiers (a, b, c) est dénombrable.

D. L'ensemble des fonctions de $\mathbb{N} \to \mathbb{N}$ est dénombrable.

E. L'ensemble des nombres rationnels est dénombrable.

A,B,C,E.

Logical Circuits: Notation

In the following questions, we will use the symbol $\cdot$ for **logical and**, $+$ for **logical or**, and $\overline{X}$ for the **negation** of X. Furthermore, we use the default order of precedence, i.e., **negation** before **and** before **or**. Figure 1 shows the symbols that we will use for AND, OR, and NOT gates.

Dans les questions suivantes, le symbole $\cdot$ indique la **conjonction** (AND/ET), le symbole $+$ indique la **disjonction** (OR/OU), et $\overline{X}$ indique la **négation** de la variable X. La priorité des opérations est **négation** avant **conjonction** avant **disjonction**. La Figure 1 montre les symboles que nous utiliserons pour indiquer les circuits AND/ET (à gauche), OR/OU (au milieu) et NOT/INVERSEUR (à droite).

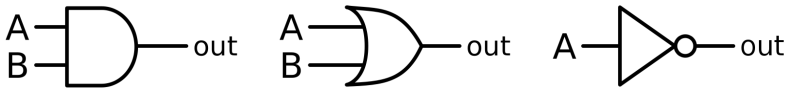

Figure 1: Symbols for AND (left), OR (middle), and NOT(right) gates

12 Question: Logical Circuits

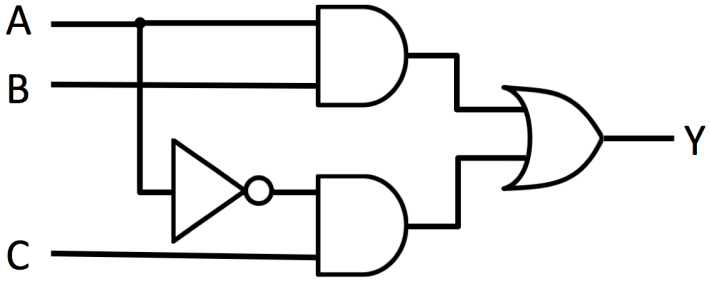

Figure 2: Circuit 1

Consider the circuit in Figure 2 and **select all correct statements.**

A. When $A = 1$ the output Y of the circuit is equal to B.

B. When $A = 0$ the output Y of the circuit is equal to C.

C. When $A = 1$ the output Y of the circuit is always 1.

D. When $A = 0$ the output Y of the circuit is always 0.

Considérez le circuit de la Figure 2 et **choisissez toutes les réponses vraies.**

A. Lorsque $A = 1$, la sortie Y du circuit est égale à B.

B. Lorsque $A = 0$, la sortie Y du circuit est égale à C.

C. Lorsque $A = 1$, la sortie Y du circuit est toujours 1.

D. Lorsque $A = 0$, la sortie Y du circuit est toujours 0.

A and B

Assembly Code: Notation

For the following two questions assume you are given a CPU with ten registers (`r0` to `r9`) and the basic instructions given in Table 1.

Pour les deux questions suivantes supposez que vous avez à disposition un CPU avec dix registres (`r0` à `r9`) et les instructions de base donnés à la Table 1.

Table 1: 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
`multiply r0, r1, r2`	$r0 \leftarrow r1 \cdot r2$: multiply the values of register r1 and r2 and put the result in r0
`divide 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 ($r0 > r1$)
`stop`	Stop the program

13 Question: Assembly Code

```
 1: copy     r2, 1
 2: copy     r3, 1
 3: copy     r4, 1
 4: copy     r5, 0
 5: add      r5, r5, 1
 6: jump_gt  r5, r1, 12
 7: multiply r3, r3, r0
 8: multiply r4, r4, r5
 9: divide   r6, r3, r4
10: add      r2, r2, r6
11: jump     5
12: stop
```

Figure 3: An assembly program

Consider the assembly program shown in Figure 3. What is the value of `r2` after executing this assembly program, if `r0` is initially 2 and `r1` is initially 3?
Considérez le code assembleur illustré à la Figure 3. Quelle est la valeur de `r2` après avoir exécuté ce programme assembleur en partant de `r0` valant 2 et `r1` valant 3 au début du programme ?

A. 5

B. 6

C. 7

D. 8

B. The assembly program computes the taylor expansion of e^x, i.e., $f(x,n) = \sum_{i=0}^{n} \frac{x^i}{i!}$, where x is stored in `r0` and n is stored in `r1`.

```
 1: copy     r2, 1           // sum = 1 (expansion of n=0)
 2: copy     r3, 1           // product = 1
 3: copy     r4, 1           // factorial = 1
 4: copy     r5, 0           // i = 0
 5: add      r5, r5, 1       // i++
 6: jump_gt  r5, r1, 12      // if (i > n) stop
 7: multiply r3, r3, r0      // product = product * x
 8: multiply r4, r4, r5      // factorial = factorial * i
 9: divide   r6, r3, r4      // quotient = product / factorial
10: add      r2, r2, r6      // sum = sum + quotient
11: jump     5               // goto line 5 (i++)
12: stop
```

14 Open Question: Assembly Code

Consider ALGORITHM4 (shown below) that takes two positive integers x and y as input. Assume x is stored in register `r0` and y is stored in register `r1`. Write an assembly program that computes the same output as ALGORITHM4 and puts it in `r2`, i.e., at the end of the execution of your assembly program register `r2` should hold the value r. Recall that you are given a CPU with ten registers (`r0` to `r9`) and the basic instructions given in Table 1.
Considérez ALGORITHM4 ci-dessous qui prend en entrée deux entiers positifs x et y. Supposons que la variable x est stockée dans le registre `r0` et y est stockée dans le registre `r1`. Ecrivez un programme assembleur qui calcule la même chose que ALGORITHM4 et qui retourne son résultat dans `r2`. Cela veut dire que le registre `r2` devrait contenir la valeur r à la fin de l'exécution de votre programme assembleur. Rappelez-vous que vous avez à disposition un CPU avec dix registres (`r0` à `r9`) supportant les instructions basiques données à la Table 1.

Algorithm 4 ALGORITHM4(x, y)

```
1: r ← 1
2: for i from 1 to y do
3:     s ← 0
4:     for j from 1 to x do
5:         s ← s + r
6:     r ← s
7: return r
```

Note that ALGORITHM4 computes x^y using only addition.

```
L1: r2 = 1;                 // copy r2, 1
L2: r3 = 1;                 // copy r3, 1
L3: if (r3 > r1) goto L13; // jump_gt r3, r1, 13
L4: r5 = 0;                 // copy r5, 0
L5: r4 = 1;                 // copy r4, 1
L6: if (r4 > r0) goto L10; // jump_gt r4, r0, 10
L7: r5 = r5 + r2;           // add r5, r5, r2
L8: r4 = r4 + 1;            // add r4, r4, 1
L9: goto L6;                // jump 6
L10: r2 = r5;               // copy r2, r5
L11: r3 = r3 + 1;           // add r3, r3, 1
L12: goto L3;               // jump 3
L13: return r2;             // stop
```

15 Open Question: Writing Algorithms

(a) Write an algorithm that takes a list as input and returns one of the elements that appears the most frequently in the list, e.g., given the list $\{4, 6, -3, 1, 4, 10, 4, 1, 2, 10\}$, 4 appears three times, the number 1 and 10 appear each twice, and all the other numbers appear only once. Therefore the algorithm should return 4. If the input list is $\{4, 6, -3, 2\}$, the algorithm is allowed to return one of the four numbers in the list, whichever you like, because all of them appear the same amount of time (namely once). The input to your

algorithm is a list L with n elements. The output is one of the elements of the list. You can assume that the list has at least one element. (In your algorithm, you can use n to refer to the size of the list L and $L[i]$ to access the i-element in the list; the first element has index 1.)

(b) Give the (asymptotic worst-case) complexity of your algorithm in terms of n (the size of the input list L) using the big O notation (Landau Notation).

(c) Finally, assume that the input list is sorted. Write an algorithm that returns the same output as in question (a) and runs in $O(n)$. You can again assume that the list has at least one element.

(a) Écrivez un algorithme qui prend une liste en entrée et qui retourne un des éléments qui apparaît le plus fréquemment dans la liste. Par exemple pour la liste d'entrée $\{4, 6, -3, 1, 4, 10, 4, 1, 2, 10\}$, 4 apparaît trois fois, les nombres 1 et 10 apparaissent chacun deux fois, et tous les autres nombres n'apparaissent qu'une seule fois. L'algorithme devrait donc retourner 4. Si la liste d'entrée est $\{4, 6, -3, 2\}$, l'algorithme a l'autorisation de retourner l'un quelconque des quatre nombres de la liste (n'importe lequel, selon votre choix), puisque tous les nombres apparaissent le même nombre de fois (une seule fois). L'entrée de votre algorithme est une liste L avec n éléments. La sortie est un des éléments de la liste. Vous pouvez supposer que la liste a au moins un élément. (Dans votre algorithme, vous pouvez utiliser n pour vous référer à la taille de la liste L et $L[i]$ pour accéder aux i-ième élément de la liste; le premier élément est à l'index 1.)

(b) Donnez la complexité (asymptotique dans le pire des cas) de votre algorithme en terme de n (la taille de liste d'entrée L) en utilisant la notation de Landau.

(c) (c) Finalement, supposez que la liste d'entrée soit triée. Écrivez un algorithme que retourne la même sortie qu'à la question (a) et qui s'exécute en $O(n)$. Vous pouvez de nouveau supposer que la liste a au moins un élément.

(a)

Algorithm 5 MOST_FREQUENT_ELEMENT(L, n)

```
1:  maxElem ← L[1]
2:  maxCnt ← 1
3:  for i from 1 to n do
4:      cnt ← 0
5:      for j from 1 to n do
6:          if L[i] = L[j] then
7:              cnt ← cnt + 1
8:      if cnt > maxCnt then
9:          maxElem ← L[i]
10:         maxCnt ← cnt
11: return maxElem
```

(b) $O(n^2)$

(c)

Algorithm 6 MOST_FREQUENT_ELEMENT_FAST(L, n)

```
1:  maxElem ← L[1]
2:  maxCnt ← 1
3:  cmp ← L[1]
4:  cnt ← 1
5:  for i from 2 to n do
6:      if L[i] = cmp then
7:          cnt ← cnt + 1
8:      if cnt > maxCnt then
9:          maxElem ← cmp
10:         maxCnt ← cnt
11:     if L[i] ≠ cmp then
12:         cmp ← L[i]
13:         cnt ← 1
14: return maxElem
```

You can use this page as scratch paper.
Vous pouvez dégrafer cette page et l'utiliser comme brouillon.

You can use this page as scratch paper.
Vous pouvez dégrafer cette page et l'utiliser comme brouillon.