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 :

- Vous disposez de 3 heures pour faire cet examen (8h00 - 11h).
- Nombre maximum de points : 100
- **Attention : il y a aussi des énoncés sur le verso.**
- Vous devez **écrire à l'encre noire ou bleue foncée**, pas de crayon ni d'autres couleurs.
- La documentation autorisée consiste en 1–3 livres de référence ainsi que quatre feuilles recto-verso de notes sur papier pour chaque partie du cours (quatre pour la théorie et quatre pour la pratique). Une version imprimée et agrafée du BOOC compte comme un livre de référence. Tout matériel électronique (y compris les calculatrices) est interdit.
- Répondez sur les feuilles qui vous sont distribuées et **aux endroits prévus**. **Ne répondez pas sur l'énoncé**.
- Ne joignez aucune feuille supplémentaire ; **seules les feuilles de réponses distribuées seront corrigées**.
- Vous pouvez répondre aux questions en français ou en anglais.
- L'examen compte 5 exercices indépendants (2 pour la théorie et 3 pour la programmation).

Vous pouvez commencer par celui que vous souhaitez

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

suite au verso ☞

Exercice 1 : QCM Partie théorie [16 points]

Pour chaque question à choix multiple vous pouvez obtenir p points. Pour des réponses partiellement correctes, vous recevrez une fraction des points selon la procédure suivante :

- $+p/N$ points si vous sélectionnez une réponse correcte et N est le nombre de réponses correctes,
- 0 points si vous ne répondez pas,
- $-p/M$ points si vous choisissez une réponse incorrecte et M est le nombre de réponses incorrectes.

QCM 1.1 : Notation de Landau [1 point]

Les affirmations suivantes utilisent différentes notations de Landau. Indiquez lesquelles de ces affirmations sont correctes. (**Plusieurs réponses correctes sont possibles.**)

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

Algorithmes et leur complexité

Considérez les algorithmes suivants ALGO1 et ALGO2. ALGO1 prend une liste L de nombres en entrée et renvoie un nombre. Le deuxième algorithme ALGO2 prend également une liste L de nombres en entrée et renvoie une liste de nombres. 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 $L[1], L[2], \ldots L[n]$. De plus, l'algorithme SIZE(L) prend la liste L en entrée et renvoie le nombre d'éléments dans L.

Algorithm 1 ALGO1(L)

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

Algorithm 2 ALGO2(L)

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

QCM 1.2 : Exécution d'un algorithme [1 point]

Lequel des nombres suivants est renvoyé lorsque ALGO1 est appelé avec la liste $\{1, 6, 8, 9, 4, 6, 2, 4, 9\}$?

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

QCM 1.3 : Exécution d'un algorithme [1 point]

Laquelle des listes suivantes est renvoyée lorsque ALGO2 est appelé avec la liste $\{1, 6, 8, 9, 4, 6, 2, 4, 9\}$?

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 : Complexité [1 point]

Soit n la longueur de la liste L, c'est-à-dire $n = \text{size}(L)$, et supposons que $T(n)$ décrit le nombre d'instructions que ALGO2(L) exécute dans le pire des cas en fonction de n. Nous supposons que le nombre d'instructions pour calculer la taille d'une liste (SIZE) est constant. Laquelle de ces affirmations est correcte ?

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 : Dénombrabilité [2 points]

Lesquelles des affirmations suivantes sont vraies ? (**Plusieurs réponses correctes sont possibles.**)

A) L'ensemble de toutes les fonctions $f : \mathbb{B} \to \mathbb{N}$ est dénombrable.
B) L'ensemble des séquences infinies de nombres $\{0 - 9\}$ est dénombrable.
C) L'ensemble des entiers non divisibles par 5 $\{z \in \mathbb{Z} \mid z \mod 5 \neq 0\}$ est dénombrable.
D) L'ensemble de tous les programmes écrits en C++ n'est **pas** dénombrable.

Algorithmes récursifs et leur complexité

Considérez l'algorithme ALGO3 ci-dessous, qui prend une liste L de nombres en entrée et renvoie une liste de nombres. Nous utilisons les notations PUSH(L, x) pour ajouter l'élément x à la fin de la liste L et POP(L) pour supprimer le dernier élément de la liste L. De plus, SIZE(L) fait référence à la taille de la liste L.

Algorithm 3 ALGO3(L)

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

QCM 1.6 : Algorithme récursif [2 points]

Laquelle des listes suivantes est renvoyée lorsque **Algorithm 3** est appelé avec l'entrée $L = \{3, 4, 1, 7, 9, 2\}$, c'est-à-dire 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 : Complexité [2 points]

Quelle est la complexité (asymptotique dans le pire des cas) de l'algorithme ALGO3 si nous supposons que l'ajout d'un élément à la fin de la liste ou la suppression d'un élément de la fin de la liste est dans $\Theta(1)$ et que n est la taille de la liste d'entrée L ?

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

QCM 1.8 : Circuits [2 points]

Pour rappel, nous utilisons les symboles illustrés dans la figure 1 pour représenter les portes logiques ET (AND), OU (OR) et NON (NOT), respectivement. Notez que deux fils qui se croisent ne sont connectés que s'il y a un petit cercle noir rempli au croisement.

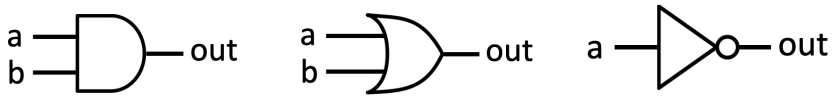

FIGURE 1 – Symboles d'une porte ET (AND, à gauche), d'une porte OU (OR, au milieu) et d'une porte NON (NOT, à droite).

Soit la table de vérité du tableau QCM 1.8 : . Sélectionnez **tous les** circuits donnés dans la Figure 2 qui calculent la fonction **f**.

TABLE 1 – table de verité de la fonction f

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

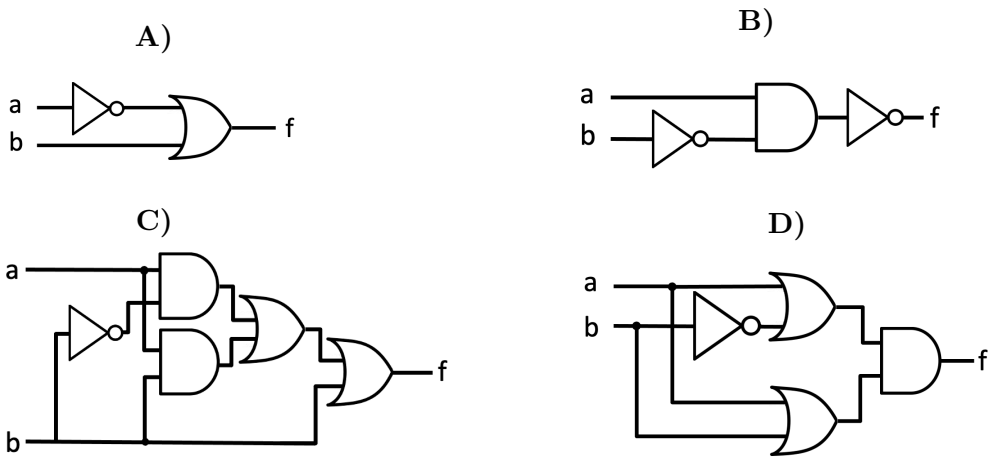

FIGURE 2 – Circuits

QCM 1.9 : Langage assembleur [2 points]

Supposons que l'on ait un CPU avec dix registres (`r0` à `r9`) et les instructions de bases données dans la Table 2.

TABLE 2 – Instructions de base

Instruction	Signification
`copy r0, c`	$r0 \leftarrow c$: copie une constante c dans le registre r0
`copy r0, r1`	$r0 \leftarrow r1$: copie la valeur du registre r1 dans le registre r0
`add r0, r1, c`	$r0 \leftarrow r1 + c$: additionne une constante c à la valeur du registre r1 et enregistre le résultat dans le registre r0
`add r0, r1, r2`	$r0 \leftarrow r1 + r2$: additionne les valeurs des registres r1 et r2 et enregistre le résultat dans r0
`sub r0, r1, c`	$r0 \leftarrow r1 - c$: soustraire la constante c du registre r1 et enregistre le résultat dans r0
`sub r0, r1, r2`	$r0 \leftarrow r1 - r2$: soustraire la valeur du registre r2 de r1 et enregistre le résultat dans r0
`mul r0, r1, r2`	$r0 \leftarrow r1 \cdot r2$: multiplie les valeurs des registres r1 et r2 et enregistre le résultat dans r0
`div r0, r1, r2`	$r0 \leftarrow r1/r2$: division entière du registre r1 par r2 ; le résultat est enregistré dans r0
`jump n`	va à la ligne n du programme
`jump_gt r0, r1, n`	va à la ligne n si la valeur dans r0 est supérieure à la valeur dans r1 ($r0 > r1$)
`jump_ne r0, r1, n`	va à la ligne n si la valeur dans r0 n'est pas égale à la valeur dans r1 ($r0 \neq r1$)
`stop`	stoppe le programme

Soit le programme en assembleur ci-contre à droite et les trois programmes ci-dessous, chacun prenant deux entiers x et y en entrée et retournant un entier. L'opérateur de barre oblique ($/$) dans les programmes fait référence à une division entière, par exemple, $5/2 = 2$. Lequel des programmes (A, B ou C) calcule le même résultat que le programme en assembleur, si x est stocké dans `r0`, y est stocké dans `r1`, et r est stocké dans `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 ← 1
2 : if y > 0 then
3 :     for i from 1 to y do
4 :         z ← 1
5 :         if (i/2)·2 = i then
6 :             z ← x
7 :         r ← r · z
8 : return r
```

B)

```
B(x, y)
1 : r ← 1
2 :
3 : for i from 1 to y do
4 :     z ← 1
5 :     if (i/2)·2 ≠ i then
6 :         z ← x
7 :     r ← r · z
8 : return r
```

C)

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

QCM 1.10 : Mémoire [2 points]

Considérons les huit premiers blocs de la mémoire vive (RAM) d'un ordinateur. Supposons que chaque bloc ait 16 mots. On s'intéresse donc aux 128 premiers mots de la RAM. Chaque bloc est indexé par l'adresse de son premier mot : $0, 16, 32, 48, 64, 80, 96, 112$. De plus, supposons que cet ordinateur ait un cache qui peut contenir jusqu'à 3 blocs, et que les blocs 16 et 96 soient déjà chargés dans le cache. Etant donné un processeur utilisant l'algorithme LRU (Least Recently Used) et qui accède à la séquence d'adresses mémoire suivante :

19 8 30 99 77 32 31 79

Combien de « cache misses » sont générés ?

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

TABLE 3 – Extrait de la table ASCII

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

Exercice 2 : Écriture d'un algorithme [20 points]

1. On vous donne deux algorithmes appelés LETTERTOINT et INTTOLETTER qui convertissent une lettre en sa représentation entière et inversement en suivant le codage ASCII donné dans Table 3, par exemple, LETTERTOINT(A) = 65 et INTTOLETTER(100) = d. Utilisez ces algorithmes pour écrire un algorithme appelé TOLOWERCASE(L) qui prend un mot donné comme une liste de lettres, par exemple, {L,a,u,s,a,n,n,e}, renvoie une autre liste dans laquelle toutes les lettres majuscules (A-Z) sont remplacées par des lettres minuscules (a-z), par exemple TOLOWERCASE({L,a,u,s,a,n,n,e}) devrait renvoyer {l,a,u,s,a,n,n,e}. (Étant donné la valeur ASCII d'une lettre majuscule, il suffit d'ajouter une valeur fixe pour obtenir la valeur ASCII de la lettre minuscule correspondante.)
2. Écrivez un algorithme appelé SORTLETTERS(L) qui prend une liste de lettres en entrée et renvoie une version alphabétiquement triée de cette liste. Toutes les lettres majuscules doivent être placées avant les lettres minuscules (comme dans la table ASCII), par exemple, SORTLETTERS({L,a,u,s,a,n, n,e}) devrait retourner {L,a,a,e,n,n,s,u}. Rappelez-vous que vous pouvez utiliser l'algorithme SORT que nous avons vu dans le cours pour obtenir une version triée d'une liste d'entiers.
3. Écrivez un algorithme appelé ISANAGRAM(L_1, L_2) qui prend deux mots en entrée et renvoie vrai si le premier mot est une anagramme du second, sinon il renvoie faux. Un mot est une anagramme d'un autre mot s'il peut être construit en réarrangeant les lettres de l'autre mot. Nous ne faisons pas de différence entre les lettres minuscules et majuscules, par exemple, « silent » est une anagramme de « LISTEN ». Chaque mot est donné sous la forme d'une liste de lettres, par exemple {s,i,l,e,n,t} et {L,I,S,T,E,N}. Vous pouvez utiliser les algorithmes TOLOWERCASE et SORTLETTERS des tâches précédentes.
4. Quelle est la complexité de ISANAGRAM(L_1, L_2) en fonction de m, où m est le nombre maximum de lettres dans le premier et le second mot, c'est-à-dire $m = \max(n_1, n_2)$, où n_i est le nombre d'éléments dans L_i. (Rappelez-vous que vous pouvez supposer que le nombre d'instructions exécutées dans SIZE(L) est égal à $\Theta(1)$ et que le nombre d'instructions exécutées dans SORT(L) est égal à $\Theta(n \cdot \log n)$, où n est le nombre d'éléments dans L).

Exercice 3 : Syntaxe et bases de C++ [15 points]

1. Pour chacune des instructions suivantes (prise séparément) indiquez si elle correspond à une instruction correcte ou incorrecte en C++. Dans les cas corrects indiquez à quoi ces instructions correspondent.

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. Soit la déclaration suivante :

```
struct Image { };
```

Quelles sont les affirmations correctes parmi les suivantes ?

1. `Image` est un type.
2. `Image` est une variable.
3. `Image` est un tableau.
4. `Image a;` est une déclaration correcte.
5. `Image a; a.pixel=2;` est un code correct.
6. `typedef Ptr Image*;` est une instruction valide.
7. `typedef Image* Ptr;` est une instruction valide.

Exercice 4 : Conception de programmes et programmation [34 points]

Vous aidez des *entreprises de garde d'animaux* («pet sitting») à organiser leur travail hebdomadaire. Chaque entreprise emploie un ensemble de gardiens («pet sitters»). Chaque *gardien* est caractérisé par son *nom* et son *emploi du temps*. Un emploi du temps indique pour chaque jour de la semaine la garde effectuée par le gardien (on considère qu'il ne peut y avoir qu'une garde par jour au maximum). Une *garde* est caractérisée par l'*adresse* de l'animal pris en charge, son *type* et s'il a dû être emmené en *promenade* ou pas. En plus de son ensemble de gardiens, une entreprise est caractérisée par son *nom*, le *tarif de base* appliqué à une garde (un *double*) et la majoration de ce tarif (exprimée en pourcentage du tarif de base) en cas de promenade (un *double*).

Vous considérerez que les types suivants sont fournis et qu'ils doivent être utilisés

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

Question 1 : Structures de données/types [12 points]

Donnez un code C++ possible pour les types/structures de données permettant de modéliser : une garde, un emploi du temps, un gardien, et une entreprise de garde d'animaux.

Les définitions des types/structures de données seront données dans l'ordre de précédence exigé par C++. Vous pourrez bien sûr définir d'autres types si nécessaire et ferez un usage judicieux du `typedef`.

Question 2 : Fonctionnalités [22 points]

Les fonctionnalités suivantes sont nécessaires à nos entreprises de gardiennage. Donnez les prototypes de ces fonctions. On ne vous demande pas d'écrire un programme complet ou le corps des fonctions mais uniquement les prototypes. Il n'est pas demandé non plus d'écrire des directives d'inclusion (`#include<..>`).

Vous devez ajoutez d'autres fonctions si vous estimez que c'est pertinent pour une **bonne modularisation**. Vous pouvez aussi **indiquer en commentaire si une fonction en utilise d'autres** parmi celles demandées si cela est nécessaire à la compréhension. Vous penserez aussi aux cas limites.

1. Afficher toutes les données d'une entreprise, ce qui inclut l'affichage de l'ensemble de ses gardiens. Pour chaque gardien seront affichés son nom, et son emploi du temps hebdomadaire. Afficher un emploi du temps revient à afficher pour chaque jour de la semaine, l'adresse de l'animal gardé, son type et si une promenade a dû se faire.
2. Ajouter un gardien à une entreprise. Ce dernier sera créé par la fonction à partir de ses données caractéristiques et avec un emploi du temps vide, ce qui signifie qu'aucun jour n'a de garde prévue.
3. Ajouter une garde à un gardien donné pour un jour donné. Cette garde sera créée par la fonction à partir de ses données caractéristiques. Si le jour en question est déjà pris par une garde, celle-ci sera simplement remplacée par la nouvelle garde.
4. Pour une entreprise, trouver sans l'afficher un gardien avec le plus grand nombre de visites dans son emploi du temps ainsi qu'un gardien avec un emploi du temps vide.
5. Pour une entreprise, trouver sans l'afficher le jour de la semaine avec le plus de gardes.
6. Pour une entreprise, trouver le type d'animal pour le quel il y a le plus grand nombre de gardes.
7. Pour une entreprise, trouver les noms des gardiens disponibles pour une garde pour un jour donné, dans une entreprise donnée.
8. Pour une entreprise, calculer les revenus de la semaine pour un gardien donné (somme des tarifs de toutes ses gardes incluant la majoration en cas de promenade).

Exercice 5 : Déroulement de programme [15 points]

Le programme suivant compile et s'exécute sans erreurs (C++11).

```
1  #include <iostream>
2  using namespace std;
3
4  typedef string Data;
5
6  struct N {
7    Data data;
8    N* left;
9    N* right;
10 };
11
12 string  p(string prefix, int rep){
13   if (rep==0) return "";
14   return (prefix + p(prefix, rep-1));
15 }
16
17 void p(N* root, int level, int depth=3)
18 {
19   if (root == nullptr) {
20     return;
21   }
22   p(root->left, level + 1, depth);
23   p(root->right, level + 1, depth);
24
25   if (depth == level) {
26     cout << root->data << " ";
27   }
28 }
29
30 N* init(Data data)
31 {
32   N* result(new N());
33   result->data = data;
34   result->left = nullptr;
35   result->right = nullptr;
36   return result;
37 }
38
39 N* c(N* n, Data left, Data right, int height){
40   if (n == nullptr || height ==0) return n;
41   n->data=left+right;
42   n->left = init(left);
43   n->right = init(right);
44   c(n->left, left, right,  height-1);
45   c(n->right, left, right, height-1);
46   return n;
47 }
```

```
48
49 int main()
50 {
51   const Data LEFT("AC");
52   const Data RIGHT("GT");
53   constexpr int  K(3);
54   N* root(init(""));
55   root = c(root, LEFT, RIGHT, K);
56   for (int i(1); i<=K; ++i){
57     cout << p(" ", K-i+1);
58     p(root, 1, i);
59     cout << endl;
60   }
61   return 0;
62 }
```

1. Que font conceptuellement les fonctions `p` ?
2. Que fait conceptuellement la fonction `c` ?
3. Quelle est la complexité de l'algorithme `c` en fonction du paramètre `height` ?. Justifiez brièvement. (Vous noterez que changer les autres paramètres n'a pas d'impact sur le nombre d'instructions exécutées dans le pire des cas.)
4. Qu'affiche le programme ? Expliquez succinctement son déroulement. Il ne s'agit pas ici de paraphraser le code, mais bien d'*expliquer* les étapes et le déroulement du programme.