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 bleu foncée**, pas de crayon ni d'autre couleur. N'utilisez **pas non plus de stylo effaçable** (perte de l'information à la chaleur).
- Toute documentation papier est autorisée ; En revanche, vous ne pouvez pas utiliser d'ordinateur personnel, ni de téléphone portable, ni aucun autre matériel électronique.
- Répondez sur les feuilles qui vous sont distribuées et **aux endroits prévus**. **Ne répondez pas sur l'énoncé**.
- Ne joignez aucune feuilles supplémentaires ; **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	20	16	15	34 (13 + 21)	15

suite au verso ☞

Exercice 1 : QCM Partie théorie [20 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 formule « $p \cdot \max(0, x_C/C - x_I/I)$ », 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. **Vous ne recevrez jamais de points négatifs pour une question.**

QCM 1.1 : Notation de Landau [2 points]

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

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

B) $n^{10} + 2^n \in O(n^{10})$

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

D) $n^3 + n + 100 \in \Theta(n^4)$

Algorithmes et leur complexité

Considérez les algorithmes suivants ALGO1 et ALGO2. ALGO1 prend deux nombres en entrée et renvoie un nombre. Le deuxième algorithme ALGO2 prend deux listes L_1 et L_2 de nombres en entrée et renvoie une liste de nombres. Nous utilisons la notation $\{\}$ pour désigner un liste vide et APPEND(L, x) pour ajouter l'élément x à la fin de la liste L. 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(a, b)

```
1 : m ← a
2 : if b < m then
3 :     m ← b
4 : return m
```

Algorithm 2 ALGO2(L_1, L_2)

```
1 : n_1 ← SIZE(L_1)
2 : n_2 ← SIZE(L_2)
3 : L ← {}
4 : m ← ALGO1(n_1, n_2)
5 : for i from 1 to m do
6 :     APPEND(L, L_1[i] + L_2[i])
7 : for i from m + 1 to n_1 do
8 :     APPEND(L, L_1[i])
9 : for i from m + 1 to n_2 do
10 :    APPEND(L, L_2[i])
11 : return L
```

QCM 1.2 : Exécution d'un algorithme [2 points]

Laquelle des listes suivantes est renvoyée lorsque **Algorithm 2** est appelé avec les listes $\{1, 2, 4, 5\}$ et $\{3, 2, 1, 5, 1, 2\}$, c'est-à-dire ALGO2($\{1, 2, 4, 5\}$, $\{3, 2, 1, 5, 1, 2\}$).

A) $\{1, 2, 4, 5, 3, 2, 1, 5, 1, 2\}$
B) $\{1, 3, 2, 2, 4, 1, 5, 5, 1, 2\}$
C) $\{4, 4, 5, 10, 1, 2\}$
D) $\{3, 2, 2, 7, 5, 7\}$

QCM 1.3 : Complexité [2 points]

Soit m le maximum des longueurs des deux listes L_1 et L_2, c'est-à-dire $m = \max(\text{size}(L_1), \text{size}(L_2))$, et supposons que $T(m)$ décrit le nombre d'instructions que ALGO2(L_1, L_2) exécute dans le pire des cas en fonction de m. Nous supposons que le nombre d'instructions pour créer une liste vide ($\{\}$), pour ajouter un élément à une liste (APPEND) et pour calculer la taille d'une liste (SIZE) est constant. Quelle affirmation est correcte ?

A) $T(m) \in \Theta(\log m)$
B) $T(m) \in \Theta(m)$
C) $T(m) \in \Theta(m^2)$
D) $T(m) \in \Theta(m^3)$

Algorithmes récursifs et leur complexité

Considérez **Algorithm 3**, qui prend deux nombres n et b en entrée et renvoie une liste de nombres. Nous utilisons les mêmes notations pour les listes ($\{\}$, APPEND et SIZE) que pour **Algorithm 2**. La barre oblique (/) fait référence à la division entière, c'est-à-dire la division dans laquelle la partie fractionnaire est supprimée, par exemple, $5/2 = 2$.

Algorithm 3 CREATELIST(n, b)

```
1 : L ← {}
2 : if n <= 1 then
3 :     APPEND(L, b + n)
4 : else
5 :     L ← CREATELIST(n/2, b)
6 :     for i from 1 to n do
7 :         APPEND(L, b + n)
8 : return L
```

QCM 1.4 : Algorithme récursif [2 points]

Laquelle des listes suivantes est renvoyée lorsque **Algorithm 3** est appelé avec les entrées $n = 4$ et $b = 3$, c'est-à-dire CREATELIST(4, 3).

A) $\{4, 5, 6, 7\}$
B) $\{4, 5, 5, 7, 7, 7, 7\}$
C) $\{4, 5, 5, 6, 6, 6, 7, 7, 7\}$
D) $\{4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7\}$

QCM 1.5 : Complexité [2 points]

Quelle est la complexité (asymptotique dans le pire des cas) de **Algorithm 3** si nous supposons que l'ajout d'un élément à une liste est en $\Theta(1)$?

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

QCM 1.6 : Computabilité [2 points]

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

A) Nous pouvons trouver un algorithme qui donne la réponse correcte pour toutes les instances du problème d'arrêt.
B) Le problème du tri d'une liste se situe dans la classe de complexité NP.
C) Nous ne savons pas si la classe de complexité P est égale à la classe de complexité NP.
D) Si un problème est dans la classe de complexité NP, alors nous pouvons trouver un algorithme en temps polynomial qui donne la bonne réponse pour toutes les instances du problème.

QCM 1.7 : Architecture matérielle [2 points]

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

A) Un ordinateur qui implémente l'architecture von Neumann peut exécuter une séquence arbitraire d'instructions tirées de l'ensemble d'instructions acceptées par cet ordinateur.
B) Tous les processeurs du monde utilisent le même ensemble d'instructions.
C) Les éléments de stockage, par exemple le pointeur d'instruction, peuvent être implémentés avec des transistors.
D) Toutes les améliorations de performances des processeurs au cours des 30 dernières années étaient dues à des transistors plus petits et plus rapides.

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.

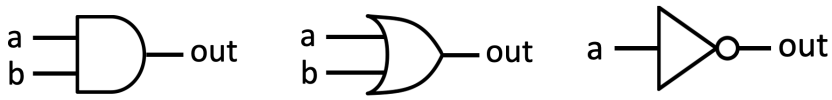

FIGURE 1 – Symboles d'une porte ET (AND, à gauche), d'une porte OU (OR, au milieu) et d'une porte NON (NOT, à droite).

Soit le circuit de la figure 2. Laquelle des tables de vérité (incomplètes) suivantes décrit la fonctionnalité de ce circuit ?

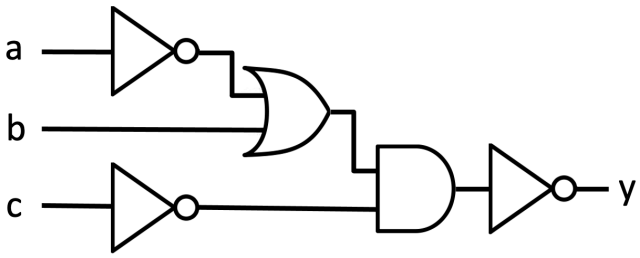

FIGURE 2 – Circuit

Table A)

a	b	c	y
0	0	0	0
..	..	..	..
1	1	0	0
1	1	1	1

Table B)

a	b	c	y
0	0	0	0
..	..	..	..
1	1	0	1
1	1	1	1

Table C)

a	b	c	y
0	0	0	1
..	..	..	..
1	1	0	0
1	1	1	1

Table D)

a	b	c	y
0	0	0	0
..	..	..	..
1	1	0	0
1	1	1	0

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 1.

Table 1 – 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_ge r0, r1, n`	va à la ligne n si la valeur dans r0 est supérieure ou égale à la valeur dans r1 ($r0 \geq r1$)
`jump_ge r0, c, n`	va à la ligne n si la valeur dans r0 est supérieure ou égale à la constante c ($r0 \geq c$)
`stop`	stoppe le programme

Soit le programme en assembleur ci-contre à droite. Quelle est la valeur de `r2` à la fin de l'exécution du programme, si `r0` est initialisé à 2 et `r1` est initialisé à 3 ?

A) 2

B) 5

C) 9

D) 14

```
1: copy r2, 0
2: copy r3, 0
3: copy r4, 0
4: jump_ge r3, r1, 9
5: add r4, r0, r3
6: add r2, r2, r4
7: add r3, r3, 1
8: jump 4
9: stop
```

QCM 1.10 : Mémoire [2 points]

Supposons que la mémoire principale d'un ordinateur (RAM) contienne 16 blocs de 2 mots chacun (32 de mots au total), et que chaque bloc soit indexé par l'adresse de son premier mot : $0, 2, 4, 6, 8, 10, \ldots, 30$. De plus, supposons que cet ordinateur dispose d'un cache pouvant contenir jusqu'à 3 blocs et qu'elle est chargée initialement avec les blocs 4 et 20. Combien d'échecs de cache se produisent lorsque le processeur utilise l'algorithme LRU (enlève le bloc le moins récement utilisé) et accède à la séquence d'adresses suivante ?

5 10 21 24 4

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

Exercice 2 : Écriture d'un algorithme [16 points]

A) Écrivez un algorithme appelé SUMROW qui, étant donné une matrice M de nombre non négatif et un index de ligne r, renvoie la somme de tous les éléments de la ligne r. La matrice est donnée sous forme de liste de listes, par exemple

$$M = \{\{1, 2, 4, 6\}, \{0, 1, 3, 1\}, \{1, 1, 1, 1\}\}$$

fait référence à une matrice à trois lignes et quatre colonnes. Si M est la matrice ci-dessus et que r vaut 2 alors l'algorithme doit renvoyer la somme de la deuxième ligne $\{0, 1, 3, 1\}$, qui est $0+1+3+1 = 5$, c'est-à-dire SUMROW($\{\{1, 2, 4, 6\}, \{0, 1, 3, 1\}, \{1, 1, 1, 1\}\}, 2$) doit renvoyer 5. Rappelons que nous utilisons la notation $L[i]$ pour accéder à l'élément i dans une liste, par exemple, $M[1]$ fait référence à la première ligne de M, qui est $\{1, 2, 4, 6\}$ et $M[1][2]$ fait référence au deuxième élément de la ligne $M[1]$, qui est 2. De plus, rappelez-vous que l'algorithme SIZE renvoie le nombre d'éléments dans une liste, par exemple, SIZE(M) vaut 3 et SIZE($M[1]$) vaut 4. La somme d'une liste vide est de 0. Nous supposons que l'index de ligne donné r est valide, c'est-à-dire $1 \leq r \leq \text{SIZE}(M)$.

B) En utilisant l'algorithme SUMROW (à partir de A), écrivez un algorithme MAXROWINDEX qui trouve l'index de la ligne avec la plus grande somme, par exemple, MAXROWINDEX($\{\{1, 2, 4, 6\}, \{0, 1, 3, 1\}, \{1, 1, 1, 1\}\}$) devrait renvoyer index 1. S'il existe plusieurs lignes avec la même somme maximale, l'algorithme doit renvoyer l'un de ces indices. De plus, si la matrice est vide, l'algorithme doit renvoyer -1.

C) Écrivez un algorithme MAXDIFF qui, étant donné une matrice non vide M de nombres non négatifs, renvoie la différence maximale entre les sommes de deux lignes quelconques, c'est-à-dire,

$$d_{\max} = \max_{i,j \in [1, \text{SIZE}(M)]} \; | \sum_{k \in [1, \text{SIZE}(M[i])} M[i][k] \; - \sum_{k \in [1, \text{SIZE}(M[j])} M[j][k] \; |.$$

Vous pouvez utiliser les algorithmes écrits en (A) et (B) et écrire des sous-algorithmes supplémentaires pour simplifier votre tâche.

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

Soit les deux déclarations suivantes faites avant `main` (vous supposerez que toutes les inclusions nécessaires sont faites) :

```
1 struct  Trip {
2   string destination;
3   size_t duration;
4 };
5
6 vector<Trip> trips(5);
```

1. Pour chacune des instructions suivantes (prise séparément) indiquez si elle correspond à une instruction correcte ou incorrecte en C++ lorsqu'elle est présente dans `main`. Justifiez brièvement les cas incorrects en indiquant si la faute est à la compilation ou à l'exécution.

a) `trips my_trips;`
b) `Trip.destination = "Karakorum";`
c) `Trip  my_trip({1, "Karakorum"});`
d) `trips[10] = {"Karakorum", 1};`
e) `Trip* my_trip(new Trip({"Karakorum", 1}));`
f) `Trip&  my_trip(new Trip({"Karakorum", 1}));`
g) `for (const auto& my_trip : trips) { cout << my_trip.destination << endl; }`
h) `for (const auto& my_trip : trips) { cout << my_trip << endl; }`
i) `for (const auto& my_trip : trips) { my_trip.destination ="Karakorum"; };`

2. Pour chacun des prototypes de fonction suivants (pris séparément), indiquez s'il est possible de l'ajouter après les deux déclarations initiales. Justifiez brièvement chacune des réponses.

a) `void f(const Trip& trip);`
b) `void f (Trip trip = {"Karakorum", 1});`
c) `Trip* f();`
d) `void f(*Trip t);`
e) `void f(Trip& t);`
f) `void f(Trip*& t);`
g) `Trip f(&Trip t);`
h) `void f(vector<Trip> trips);`
i) `void f(vector<trips> trips);`

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

Une agence de voyage vous sollicite pour l'aider à gérer ses catalogues d'offres.

Une *agence de voyage* propose des catalogues d'offres de voyages. Une *offre* est caractérisée par une destination (une `string`), son prix en basse saison (un `double`), la majoration à appliquer à ce prix en haute saison et une information indiquant si la destination nécessite un vol longue distance ou pas. On supposera que la destination est unique. Un *catalogue* est est un ensemble d'*offre*s. L'agence souhaite consigner : ses *catalogues stockés par année*[1], l'ensemble de ses *ventes en basse saison* et l'ensemble de ses *ventes en haute saison*. On considérera que ces informations caractérisent l'agence de voyage. Une *vente* est une *paire* qui associe une *date* à une *offre* de voyage. Une *date* est caractérisée par un jour, un mois et une année. Une *vente* sera considérée comme faite en haute saison si son mois est compris entre juin et septembre.

Question 1 : Structures de données/types [13 points]

Donner un code C++ possible pour les types/structures de données permettant de modéliser : une date, une offre de voyage, un catalogue d'offres, une vente, et une agence de voyage.

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 [21 points]

Les fonctionnalités suivantes sont nécessaires à notre agence. 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. créer une offre de voyage à partir de sa destination et de toutes les autres informations qui la caractérisent ;
2. calculer le prix d'une offre selon qu'elle serait achetée en haute ou basse saison ;
3. calculer le nombre de ventes faites soit en basse soit en haute saison ;
4. trouver, sans l'afficher, la destination la moins vendue pour une année donnée ;
5. ajouter une nouvelle vente (qui sera mise dans la bonne liste selon son mois) ;
6. trouver, sans l'afficher, une offre écologique pour une destination donnée dans le catalogue d'une année donnée ; une offre est écologique si elle ne nécessite pas un vol longue distance ;
7. calculer le montant des ventes écologiques (somme de toutes les ventes quelque soit la saison) ;
8. trouver la destination écologique la plus vendue en basse saison et celle la plus vendue en haute saison.

1. par exemple le catalogue de l'année 2023, celui de 2022 etc.

Exercice 5 : Déroulement de programme [15 points]

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

```
1  #include <iostream>
2  #include <vector>
3  #include <array>
4  #include <string>
5  using namespace std;
6
7  struct N {
8    vector <int> o;
9    vector <int> p;
10 };
11 typedef array<int, 3> Row;
12 typedef array<Row, 3> Set;
13
14 typedef string Str;
15
16 int c(const Str& str, char car){
17   int r(0);
18   for (auto c : str) {
19     if (c == car) ++r;
20   }
21   return r;
22 }
23 void p (const Row& r) {
24   for (auto e : r ) cout << e << " ";
25 }
26
27 void p (const Set& s){
28   for (auto e : s ) {
29     p(e);
30     cout << endl;
31   }
32 }
33
34 void p(const N& n){
35   for (auto e : n.o) cout << e << " ";
36   cout << endl;
37   for (auto e : n.p) cout << e << " ";
38   cout << endl;
39 }
40
41 int a(const Row& row, int i=0){
42   if (i == row.size()) return 0;
43   return row[i] + a(row, i+1);
44 }
45
46
47
```

```
48
49 Set i(const string& st){
50   Set s;
51   size_t w(s.size());
52   size_t h(s[0].size());
53    for (size_t i(0); i < w; ++i){
54     for (size_t j(0); j < h; ++j){
55       s[i][j] = c(st, st[i * h + j]);
56     }
57   }
58    return s;
59 }
60
61 N g(const Set& s){
62   N n;
63   for (auto r : s){
64     int val(a(r));
65     if (val%2 == 0) n.p.push_back(val);
66     else  n.o.push_back(val);
67   }
68   return n;
69 }
70
71 int main() {
72   Str st("AAGGGTTAGCCC");
73   Set s(i(st));
74   cout << "" << endl;
75   p(s);
76   cout << "" << endl;
77   p(g(s));
78
79   Set* pt(&s);
80   cout << "" << endl;
81   p(g(*pt));
82
83   return 0;
84 }
```

1. Que fait conceptuellement la fonction `c` ?
2. Que fait conceptuellement la fonction `a` ?
3. Que fait conceptuellement la fonction `g` ?
4. Qu'affiche le programme ? Expliquez succinctement sont déroulement. Il ne s'agit pas ici de paraphraser le code, mais bien d'*expliquer* les étapes et le déroulement du programme.