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 :

- 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 7 exercices indépendants.

Vous pouvez commencer par celui que vous souhaitez

Exercice	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]

Pour chaque question à choix multiple vous pouvez obtenir deux points. Pour des réponses partiellement correctes, vous recevrez une fraction des points selon la formule « $2 \cdot \max(0, x_C/C - x_I/I)$ », 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 point négatifs pour une question.**

QCM 1.1 : Complément à deux

Supposons que soit utilisée une représentation des nombres en complément à deux sur 8 bits. Quel est le résultat de l'opération $-96 - 64$ (écrit en décimal) ?

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

QCM 1.2 : Représentation en virgule flottante

Soit une représentation simplifiée en virgule flottante d'un nombre positif sur 6 bits, réalisée de la manière suivante : 3 bits pour l'exposant en base 2, 3 bits pour la mantisse. Nous utilisons la représentation des nombres en complément à deux pour l'exposant. Sélectionnez toutes les affirmations correctes :

A) Le nombre décimal 2.5 peut être représenté exactement (sans erreurs d'arrondi).
B) L'erreur relative maximum est inférieure ou égale à $2^{-4} = \frac{1}{16}$.
C) En utilisant cette représentation, on peut exactement représenter 64 nombres (sans erreurs d'arrondi).
D) Le plus grand nombre qui peut être représenté exactement (sans erreur d'arrondi) est 16.

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

Sélectionnez toutes les affirmations correctes :

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

Considérons les huit premiers blocs de la mémoire vive (RAM) d'un ordinateur. Supposons que chaque blocs ait deux mots. On s'intéresse donc aux 16 premiers mots de la RAM. Chaque bloc est indexé par l'adresse de son premier mot : $0, 2, 4, 6, 8, 10, 12, 14$. De plus, supposons que cet ordinateur ait un cache qui peut contenir jusqu'à 3 blocs, et que les blocs 2 et 4 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 :

3 6 4 10 13 5 7 3

Combien de « cache misses » sont générés ?

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

QCM 1.5 : Échantillonnage de fréquence pour la reconstruction

Supposons que l'on veuille échantillonner le signal $X(t) = \sin(34\pi t + \pi/2) + 10 \cdot \sin(18\pi t + \pi) + 2 \cdot \sin(12\pi t)$. Laquelle parmi les fréquences f_e suivantes est la plus petite fréquence qui permet une reconstruction correcte du signal $X(t)$ en utilisant la formule d'interpolation $X_I(t) = \sum_n X(nT_e)\text{sinc}(\frac{t-nT_e}{T_e})$ avec $T_e = 1/f_e$.

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

Supposons que l'on échantillonne le signal $X(t) = 2 \cdot \sin(6\pi t) + 4 \cdot \sin(12\pi t)$ avec la fréquence d'échantillonnage $f_e = 10$ Hz et que nous reconstruisons le signal $X(t)$ en utilisant la formule d'interpolation $X_I(t) = \sum_n X(nT_e)\text{sinc}(\frac{t-nT_e}{T_e})$ avec $T_e = 1/f_e$.

Parmi les affirmations suivantes, lesquelles sont correctes ?

A) En reconstruisant le signal, on observe un effet stroboscopique.

B) Le signal reconstruit $X_I(t)$ aura les mêmes fréquences que $X(t)$.

C) La fréquence apparente dans $X_I(t)$ est 4 Hz.

D) Le signal reconstruit $X_I(t)$ aura les fréquences suivantes : 2 Hz, 3 Hz et 6 Hz.

Exercice 2 : Filtre et Spectre [6 points]

Supposons que le signal suivant $X(t)$ est passé à travers un filtre passe-bas idéal avec une fréquence de coupure de 10Hz.

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

Dessinez (sur la feuille réponse) le spectre du signal sortant $\hat{X}(t)$ de ce filtre passe-bas idéal.

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

Supposez que vous vouliez envoyer la phrase suivante (sans les espaces) à votre ami en utilisant le nombre minimal de bits.

EVA CAN I SEE BEES IN A CAVE

Pour créer un encodage optimal avec l'algorithme de Huffman et trouver le nombre minimal de bits, procédez de la manière suivante :

3.1 Créez un tableau avec le nombre d'occurences pour chaque lettre (voir le tableau sur la feuille réponse.)

3.2 Dessinez un arbre de Huffman pour ce tableau.

3.3 Créez un tableau avec un encodage pour chaque lettre (voir le tableau sur la feuille réponse.)

3.4 Calculez le nombre de bits nécessaires pour envoyer ce message avec votre encodage.

3.5 Combien de bits est-ce que l'on économise avec cet encodage par rapport à un encodage ASCII qui utilise 8 bits par lettre ?

Exercice 4 : Algorithme et complexité [15 points]

Pour les questions suivantes, rappelez-vous que nous utilisons la notation $L[i]$ pour accéder au i-ème élément d'une liste L. Nous commençons à compter à 1, donc L contient les éléments $L[1], L[2], \ldots$. De plus, nous supposons que la comparaison, l'addition, la soustraction, la multiplication, la division, le modulo et l'accès à un élément d'une liste s'exécutent en temps $\Theta(1)$.

4.1 : Complexité [4 points]

Soit ALGORITHM2 qui prend en argument une liste de nombres SCIPER d'un cours EPFL ainsi que n, la taille de cette liste. L'algorithme affiche le nombre d'étudiants des groupes «Corona» A, B ou C ; c'est à dire le nombre des étudiants ayant un SCIPER qui est égal à 0, 1 ou 2 modulo 3. Afin de calculer le nombre d'étudiants dans un des groupes A, B ou C, ALGORITHM2 appelle ALGORITHM1. Ce dernier prend en argument une liste L de nombres SCIPER, la taille n de cette liste ainsi que le résultat r du calcul modulo 3 voulu. Il retourne le nombre d'étudiants dont le SCIPER est égal à r modulo 3. Quelle est la complexité (asymptotique dans le pire des cas) de ALGORITHM2 en fonction de n ?

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]

Pour calculer l'entropie d'un message m, il faut calculer pour chaque lettre l qui apparaît dans le message, la probabilité p de choisir la lettre l si l'on sélectionne uniformément un caractère au hasard dans le message.

Écrivez un algorithme nommé `compute_letter_probability` qui prend en argument

1. un message, donné sous la forme d'une liste de lettres, par exemple `message` $= \{A, B, A, B, C\}$,
2. un nombre entier qui décrit la longueur du message, par exemple `n` $= 5$, et
3. une lettre, par example `letter` $= A$

et renvoie la probabilité p. Par exemple, avec les arguments `message` $= \{A, B, A, B, C\}$, `n` $= 5$, et `letter` $= A$, l'algorithme devrait renvoyer 0.4. Avec les arguments `message` $= \{A, B, A, B, C\}$, `n` $= 5$, et `letter` $= C$, l'algorithme devrait renvoyer 0.2.

Quelle est la complexité (asymptotique dans le pire des cas) de votre algorithme `compute_letter_probability` en terme de n, en utilisant la notation Θ ?

Exercice 5 : Programmation[10 points]

Un nombre n est dit *abondant* si la somme de ses diviseurs stricts (c'est à dire ne l'incluant pas) est supérieure au n.

1. Ecrivez une fonction C++ `isDivisor` prenant en argument deux nombres et retournant `true` si le second divise le premier et ne lui est pas égal (et `false` sinon).
2. Ecrivez une fonction C++ `isAbundant` prenant en paramètre un nombre et retournant `true` s'il est abondant et `false` sinon. Cette fonction devra utiliser la fonction `isDivisor`.
3. Ecrivez une fonction `findAbundant` prenant en paramètre un entier n et retournant `true` si au moins un nombre abondant est trouvé parmi tous ceux inférieurs strictement à n. Le nombre n aura 50 comme valeur par défaut. Par ailleurs, le nombre de nombres abondants trouvés sera aussi mis à disposition par la fonction, selon l'exemple d'exécution suivant :

```
int k(0);
constexpr int upper(100);
if (findAbundant(k, upper)){ // si upper n'est pas spécifié il vaut 50
   cout << "Il existe " << k << " nombres abondants avant "
        << upper << endl;
}
```

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

Des écoles de patinage sur glace font appel à vous pour concevoir un utilitaire de gestion.

Une *école de patinage* est caractérisée par son nom et son adresse (des chaînes de caratères), par les *instructeurs* qu'elle emploie ainsi que par les *élèves* qui y sont inscrits.

Chaque élève est caractérisé par son *nom* (une chaîne de caractères) et sa *catégorie d'âge* (adulte ou enfant).

Les leçons sont données le matin, l'après-midi ou le soir tous les jours de la semaine sauf le dimanche.

Chaque instructeur est caractérisé par son nom et son planning hebdomadaire (que l'on supposera réinitialisé chaque fin de semaine). Ce *planning* sera donné par un tableau ressemblant à ceci :

	0 (matin)	1 (après-midi)	2 (soir)
0 (lundi)	*e1*		
1 (mardi)			
2 (mercredi)		*e2*	
3 (jeudi)			
4 (vendredi)			*e3*
5 (samedi)			

Chaque case du tableau devra contenir un information permettant de savoir quel élève est pris en charge par l'instructeur pour un jour donné et une période donnée. On suppose que un seul élève est pris en charge par un instructeur pour une période donnée (matin, après-midi ou soir). Dans l'exemple ci-dessus, l'instructeur donne un cours le lundi matin à l'élève *e1*, le mercredi après-midi à l'élève *e2* et le vendredi soir à l'élève *e3*. Le reste du temps il ne travaille pas.

Les alias de types suivants devront obligatoirement être utilisés dans votre conception.

```
// pour les jours de la semaine :  0 (lundi) à 5 (samedi)
typedef short int Day;
// pour les périodes de formation par les instructeurs : 0 (matin), 1 (après-midi), 2 (soir)
typedef short int TimeSlot;
```

Question 1 : structures de données [8 points]

Donner un code C++ possible pour les types/structures de données permettant de modéliser :

- Un élève.
- Une catégorie d'âge.
- Un instructeur.
- Un planning hebdomadaire.
- Une école de patinage.

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`.

Pour la modélisation du planning réfléchissez bien au type à utiliser pour le contenu des cases du tableau. Vous indiquerez en commentaire comment est représentée une case «sans élève».

Question 2 : fonctionnalités [15 points]

Les fonctionnalités suivantes sont nécessaires à la gestion de nos écoles de patinage. 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. Vous pouvez ajoutez d'autres fonctions si vous estimez que c'est pertinent pour une bonne modularisation. Il n'est pas demandé non plus d'écrire des directives d'inclusion (`#include<..>`).

1. ajouter un instructeur de nom donné à une école donnée (cette fonction sera en charge de créer l'instructeur) ;
2. inscrire un élève à une école donnée (cette fonction sera en charge de créer l'élève) ;
3. trouver sans l'afficher la catégorie d'âge la plus représentée parmi les élèves d'une école donnée ;
4. afficher les données relatives à un instructeur (ce qui implique d'afficher son nom et son planning ; l'affichage du planning affichera le nom des élèves impliqués) ;
5. trouver sans les afficher toutes les périodes libre d'un instructeur de nom donné pour un jour donné (cette fonction permettra par exemple de savoir que l'instructeur "Félicien" est libre le matin et le soir du lundi) ;
6. trouver sans l'afficher, le nombre de périodes où un instructeur donné est occupé (dénombre les cases «non vides» de son planning) ;
7. trouver sans les afficher tous les instructeurs ayant au moins une période libre pour un jour donné ;
8. affecter un élève donné à un instructeur donné, un jour donné et pour une période donnée ; cette fonction devra indiquer si l'affectation a pu avoir lieu (ce ne sera pas le cas si la case correspondante de son planning est déjà prise) ;
9. trouver sans l'afficher l'instructeur le plus populaire (celui ayant le plus de case occupée dans son planning ; on supposera que s'il y en a plusieurs seul le premier trouvé sera retourné) ;

Question 3 : programmation [7 points]

1. Donnez le code mettant en œuvre la fonctionnalité 9.

Vous utiliserez les fonctionnalités prototypées sans avoir besoin d'en donner les corps. Veillez à respecter une bonne indentation.

Exercice 7 : Déroulement de programme [15 points]

Le programme suivant compile et s'exécute sans erreurs (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. Que fait conceptuellement la fonction `f` ? quel algorithme connu y est mis en oeuvre ?
2. Quelle est la complexité temporelle de la fonction `f` en fonction de la taille `n` de la séquence `g` ?
3. 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.
4. Quelle est la complexité temporelle en `n` de la fonction `o` de la ligne 49, si l'on présupose pour simplifier que l'ensemble `gs` contient toujours `n` lignes et que chaque ligne est une chaîne de longueur `n`. On supposera que la complexité de l'algorithme `sort` est $\Theta(n \log(n))$. Justifiez brièvement.