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 une heure quarante cinq minutes pour faire cet examen (17h15 - 19h).

— Nombre maximum de points : 75.

— Toute documentation est autorisée.

— Répondez sur les feuilles qui vous sont distribuées à cet effet (utilisez aussi le verso des feuilles lorsque nécessaire). **Ne répondez pas sur l'énoncé**.

— Vous pouvez répondre aux questions en français ou en anglais.

— Veillez à **traiter chaque question à l'endroit qui lui est dédié.**

— L'examen compte 5 exercices indépendants.

**Vous pouvez commencer par celui que vous souhaitez**

— Exercice 1 : **16** points.
— Exercice 2 : **6** points.
— Exercice 3 : **8** points.
— Exercice 4 : **35** points.
— Exercice 5 : **10** points.

**suite au verso ☞**

# Exercise 1 : Théorie, question à choix multiples. [16 points]

## 1.1 Question : Circuits logiques

On rappelle qu'on utilise les symboles sur la Figure 1 pour représenter les portes logiques AND, OR, et NOT, respectivement.

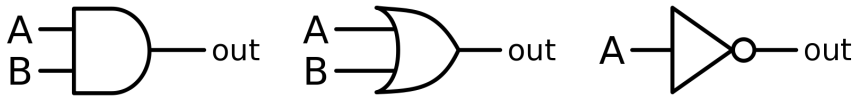



FIGURE 1 – Symboles pour une porte AND (gauche), une porte OR (milieu) et une porte NOT (droite).

Considérez le circuit sur la Figure 2 et **sélectionnez toutes les affirmations correctes.**

**A)** Si $A = 0$ et $B = 0$ la sortie $Y$ du circuit est toujours 0.

**B)** Si $B = 1$ et $C = 1$ la sortie $Y$ du circuit est égale à $D$.

**C)** Si $C = 0$ et $D = 0$ la sortie $Y$ du circuit est toujours 0.

**D)** Si $D = 0$ et $A = 1$ la sortie $Y$ du circuit est égale à $C$.

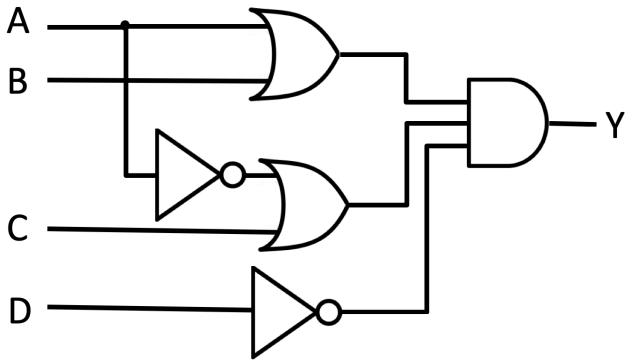



FIGURE 2 – Circuit 1

## 1.2 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 0 et 2 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 7 6 10 13 7 11 0

Combien de «cache misses »sont générés ?

**A)** 3

**B)** 4

**C)** 5

**D)** 6

### 1.3 Question : langage assembleur

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** |
|---|---|
| hline `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 |
| `multiply r0, r1, r2` | $r0 \leftarrow r1 \cdot r2$ : multiplie les valeurs des registres r1 et r2 et enregistre le résultat dans r0 |
| `divide 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 ($r0 > r1$) |
| `stop` | stoppe le programme |

Considérez 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 1 ?

**A)** 2

**B)** 4

**C)** 6

**D)** 8

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

### 1.4 Routage

Considérez le réseau de routeurs Internet donné dans la Figure 3.

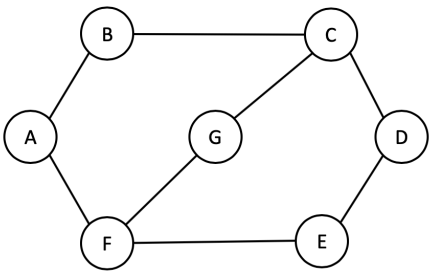



FIGURE 3 – Réseau avec 7 noeuds (A-G)

Quelle(s) affirmation(s) est(sont) vraie(s) parmi les affirmations suivantes ?

**A)** Si le noeud B est en panne, dans les tables de routage des noeuds restants, aucune distance ne sera supérieure (strictement) à 3.

**B)** Si on ajoute un lien entre B et G, la table de routage de A doit être modifiée.

**C)** La table de routage de C est

| Destination | Direction | Distance |
|---|---|---|
| A | B | 2 |
| B | B | 1 |
| D | D | 1 |
| E | D | 2 |
| F | G | 2 |
| G | G | 1 |

**D)** S'il y a un nouveau lien entre B et D, la table de routage de D contiendra la ligne suivante :

| Destination | Direction | Distance |
|---|---|---|
| A | F | 3 |

### 1.5 Moving average filter

Soit $X(t)$ une somme de sinusoïdes de bande passante $f$. X est lui-même l'entrée d'un filtre à moyenne mobile avec une fréquence de coupure $f_c$. Que peut-on dire du signal en sortie $\hat{X}(t)$ de ce filtre à moyenne mobile ?

**A)** Si $f < f_c$, alors $X(t) = \hat{X}(t)$.

**B)** Si $f < f_c$, alors $X(t)$ et $\hat{X}(t)$ ont la même bande passante.

**C)** Si $f = f_c$, alors la bande passante de $\hat{X}(t)$ est strictement inférieure à la bande passante de $X(t)$.

### 1.6 Echantillonage de fréquence pour la reconstruction

Supposons que l'on veuille échantillonner le signal $X(t) = 4 \cdot \sin(8\pi t) + 8 \cdot \sin(16\pi t + \pi) + 32 \cdot \sin(12\pi t)$. 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)** $f_e = 16$ Hz

**B)** $f_e = 17$ Hz

**C)** $f_e = 32$ Hz

**D)** $f_e = 33$ Hz

### 1.7 Effet stroboscopique

Supposons que l'on échantillonne le signal $X(t) = \sin(2\pi t) + 4 \cdot \sin(8\pi t)$ avec la fréquence d'échantillonnage $f_e = 6$ Hz. Parmi les affirmations suivantes, lesquelles sont correctes ?

**A)** La fréquence d'échantillonnage est assez grande pour permettre une reconstruction correcte du signal.

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

**C)** La fréquence apparente est 1 Hz.

**D)** Le signal reconstruit aura les deux fréquences suivantes : 1 Hz et 2 Hz.

### 1.8 Entropie

Considérez les quatre mots suivants :

$w_1 =$ `blackjacking`
$w_2 =$ `jackhammered`
$w_3 =$ `jeopardizing`
$w_4 =$ `puzzleheaded`

Parmi les affirmations suivantes, lesquelles sont correctes ?

**A)** $H(w_1) = H(w_2)$ (c'est-à-dire les mots `blackjacking` et `jackhammered` ont la même entropie)
**B)** $H(w_3) < H(w_4)$
**C)** $H(w_4) < H(w_3)$
**D)** $H(w_1) < H(w_4)$
**E)** $H(w_1) \geq H(w_3)$
**F)** $H(w_1) = H(w_4)$

## Exercise 2 : Spectre [4 points]

Dessinez (sur la feuille réponse) le spectre du signal suivant :

$$X(t) = \sin(\pi t) + 2 \cdot \sin(5\pi t) + 5 \cdot \sin(8\pi t - \pi/2) + \sin(10\pi t + \pi)$$

## Exercise 3 : Codage de Huffman's [10 points]

Supposez que vous vouliez envoyer la phrase suivante (avec les espaces) à votre ami en utilisant le nombre minimal de bits. On utilise le caractère ␣ pour les espaces.

WAS␣IT␣A␣CAT␣I␣SAW

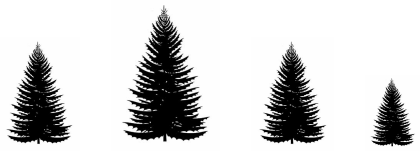
Pour créer un encodage optimal avec l'algorithme de Huffman et trouver le nombre minimal de bits, procédez de la manière suivante :

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.

**suite au verso ☞**

# Exercice 4 : Conception de programme et programmation [35 points]

Les domaines forestiers de votre canton souhaitent que vous les aidiez à gérer leurs ventes de sapins pour les fêtes de fin d'année.



Un domaine dispose de plusieurs *terrains* et est attaché à une ville, identifiée par un *code postal*. Chaque terrain est caractérisé par son nom (par exemple *"Sapinière d'Ecublens"*, *"Sapinière de Chavannes"* etc.), sa surface, l'ensemble des sapins qui y sont plantés et l'ensemble des sapins qui y ont étés vendus. Chaque terrain a par ailleurs connaissance du domaine auquel il appartient.

Un *sapin* est caractérisé par sa *variété* (une chaîne de caractères telle que *"Epicéa standard"*, *"Nordmann"* etc. ) et la *date* à laquelle il a été planté. La hauteur du sapin se calcule en fonction de cette date. Le calcul du prix de vente du sapin dépend quant à lui de sa hauteur et de sa variété. Les modalités précises de ces calculs ne nous intéressent pas, mais vous supposerez que pour chacun d'eux les traitements à réaliser sont relativement complexes.

## Question 1 : structures de données [9 points]

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

1. Un sapin.
2. Un terrain.
3. Un domaine.

Les définitions des 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 [18 points]

Les fonctionnalités suivantes sont nécessaires à la gestion de la vente des sapins :

1. calculer le prix d'un sapin donné ;
2. ajouter un terrain donné à un domaine (le terrain devra alors être attaché à ce domaine) ;
3. afficher un domaine donné (ce qui implique d'afficher son nom et les caractéristiques de chaque terrain ; à savoir son nom, sa surface, le nombre de sapins disponibles et le nombre de sapins vendus) ;
4. vendre un sapin de hauteur supérieure ou égale à une hauteur donnée et d'une variété donnée, dans un terrain donné (ce qui doit le faire passer de l'ensemble des sapins à vendre à l'ensemble des sapins vendus) ;
5. planter un sapin donné (à vendre) dans un terrain de nom donné, d'un domaine donné ; ce traitement retournera le prix total des sapins à vendre suite à l'ajout de ce sapin ;
6. calculer le montant des ventes d'un terrain de nom donné, d'un domaine donné ;
7. calculer le montant des ventes d'une variété particulière de sapin dans un terrain de nom donné, d'un domaine donné ;
8. retourner tous les terrains pour lesquels le montant des ventes est inférieur à un certain seuil donné, pour un domaine donné ;

9. trouver (sans l'afficher) quel terrain d'un domaine donné contient un sapin (à vendre) de prix inférieur à un montant donné, de variété donnée, et retourner ce sapin (il suffit de trouver un terrain et un sapin, même s'il y en a plusieurs).

Donnez le prototype de ces fonctions. Il vous est imposé la contrainte que les fonctionnalités 6 et 7 soit mises en oeuvre au moyen d'une même fonction.
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<..>`).

### Question 3 : programmation [8 points]

1. Donnez le code mettant en œuvre la fonctionnalité 2.
2. 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.

**suite au verso ☞**

## Exercice 5 : Déroulement de programme [10 points]

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

```
0.  #include <iostream>
1.  #include <vector>
2.  #include <array>
3.  #include <utility>
4.
5.  using namespace std;
6.
7.  struct Sem {
8.    string nom;
9.    vector<double> res;
10. };
11.
12. double r(vector<double> res)
13. {
14.    if (res.size() == 0) return 0.0;
15.    if (res.size() == 1) return res[0];
16.    double last_val(res[res.size()-1]);
17.    res.pop_back();
18.    return (last_val + r(res));
19. }
20.
21. void t(array<Sem, 4>& sems, bool d)
22. {
23.    const size_t l(sems.size()-1);
24.    for (size_t i(0); i < l; ++i) {
25.      for (size_t j(l); j > i; --j) {
26.         double val1(r(sems[j-1].res));
27.         double val2(r(sems[j].res));
28.         if ((d and (val1 < val2) or
29.            (not d and (val1 > val2)))) {
30.            swap(sems[j-1], sems[j]);
31.         }
32.      }
33.    }
34. }
```

```
35. void d(const Sem& s)
36. {
37.    cout << "("
38.       << s.nom
39.       << ':'
40.       << r(s.res)
41.       << ") "
42.       << flush;
43. }
44.
45. void d(const array<Sem, 4>& sems)
46. {
47.    for (const auto& sem : sems) {
49.      d(sem);
48.    }
50.    cout << endl;
51. }
52.
53. int main()
54. {
55.    array<Sem, 4>  s = { "s1", { 5.0, 6.0 },
56.                    "s2", { 2.0, 6.0 },
57.                    "s3", { 1.0, 2.0 },
58.                    "s4", { 6.0, 6.0, 5.0 }
59.                  };
60.
61.    d(s[0]);
62.    cout << endl;
63.    cout << r(s[0].res) << endl;
64.    d(s[0]);
65.    cout << endl;
66.    cout << "@@@@" << endl;
67.    d(s);
68.    t(s, true);
69.    d(s);
70.    t(s, false);
71.    d(s);
72.    return 0;
73. }
```

1. Qu'affiche t-il ? 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.
2. Que font conceptuellement les fonctions `r` et `t` ?