EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours d'Information, Calcul et Communication, sections MA et PH
Chappelier J.-C.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

NOM : Hanon Ymous
(000000) #0000
Place : 0

Information, Calcul et Communication (SMA/SPH) : Examen III

21 décembre 2018

SUJET 1

INSTRUCTIONS (à lire attentivement)

IMPORTANT ! Veuillez suivre les instructions suivantes à la lettre sous peine de voir votre examen annulé dans le cas contraire.

1. Vous disposez de deux heures quarante-cinq minutes pour faire cet examen (13h15 – 16h00).
2. 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).
3. Vous avez droit à toute documentation papier.
   En revanche, vous ne pouvez pas utiliser d'ordinateur personnel, ni de téléphone portable, ni aucun autre matériel électronique.
4. Répondez aux questions directement sur la donnée, **MAIS** ne mélangez pas les réponses de différentes questions !
   Ne joignez aucune feuilles supplémentaires ; **seul ce document sera corrigé**.
5. Lisez attentivement et *complètement* les questions de façon à ne faire que ce qui vous est demandé. Si l'énoncé ne vous paraît pas clair, ou si vous avez un doute, demandez des précisions à l'un des assistants.
6. L'examen comporte 6 exercices indépendants sur 20 pages, qui peuvent être traités dans n'importe quel ordre, mais qui ne rapportent pas la même chose (les points sont indiqués, le total est de 150 points) :
   1. questions courtes : 26 points ;
   2. problème : 63 points ;
   3. mélange de mots : 12 points ;
   4. compréhension de programme : 16 points
   5. la cache est déterminante : 19 points et
   6. chercher les erreurs : 14 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Questions courtes [26 points]

Question 1.1 [4 points]

Un sac contient 24 billes de 4 couleurs différentes : 12 billes rouges, 8 billes bleues, 2 billes vertes, et 2 billes oranges. Quelle est, en bit, l'entropie du jeu consistant à deviner la couleur d'une bille tirée au hasard ?

Donnez votre réponse sous la forme $a + b \log_2(3)$ (précisez les valeurs de a et b, éventuellement sous forme de fraction) :

Question 1.2 [5 points]

A partir d'un alphabet de 33 lettres, on compose un mot X de 128 lettres au total, chacune des lettres de l'alphabet étant présente au moins une fois dans le mot X. Le code de Huffman de ce mot a une longueur moyenne de 5.5 bits.

1. Est-ce possible ? Justifiez *brièvement* votre réponse.

2. Si **oui**, donnez les *meilleures* bornes possibles (haute et basse) que vous pouvez pour
   a) l'entropie de ce mot :
   $$\leq H(X) \leq$$
   b) la longueur moyenne d'un code de Shannon-Fano de ce mot :
   $$\leq L_c(\text{Shannon-Fano}(X)) \leq$$
   et si c'est **non**, donnez les *meilleures* bornes possibles (haute et basse) que vous pouvez pour la longueur moyenne d'un code de Huffman de ce mot :
   $$\leq L_c(\text{Huffman}(X)) \leq$$
   Justifiez brièvement vos réponses.

Question 1.3 [5 points]

Des séquences de cinq niveaux d'alerte météo doivent être transmises codées (code sans-préfixe et sans perte) sous forme de séquences de pastilles (ronds) rouges ou vertes. La table ci-dessous représente trois propositions de codes possibles. Malheureusement, ce sujet est tiré en noir et blanc ; la couleur verte ou rouge s'est donc perdue...

code I	code II	code III
niveau 1 : ●	niveau 1 : ●●	niveau 1 : ●●●
niveau 2 : ●●●●	niveau 2 : ●●●	niveau 2 : ●
niveau 3 : ●●	niveau 3 : ●●●	niveau 3 : ●●●
niveau 4 : ●●●●	niveau 4 : ●●	niveau 4 : ●
niveau 5 : ●●	niveau 5 : ●●	niveau 5 : ●●

Quel(s) code(s) (d'origine, avec les couleurs) êtes vous néanmoins sûr(e) de ne pas pouvoir utiliser pour la transmission désirée ? Justifiez *brièvement* votre réponse.

Question 1.4 [7 points]

Soit f une fonction de $\mathbb{R}$ dans $\mathbb{R}$ définie par

$$f(t) = \sum_{m\in\mathbb{Z}} a_m \,\mathrm{sinc}(30\,t - m)$$

où $\mathrm{sinc}(x) = \dfrac{\sin(\pi\,x)}{\pi\,x}$ et $a_m = 7\sin(\dfrac{2\pi}{3}m + \dfrac{\pi}{6}) + 3\sin(\dfrac{4\pi}{5}m + \dfrac{\pi}{4}) + 2\sin(\dfrac{2\pi}{6}m)$.

Écrire $f(t)$ comme la somme de trois fonctions sinus :

$$f(t) =$$

Justifiez brièvement votre réponse.

suite au dos ☞

Question 1.5 [5 points]

Quel(s) problème(s) existant(s) ou potentiel(s) voyez vous dans l'*extrait* de code suivant :

```
int s(0);
for (int n(0); n <= v.size(); ++n) {
    s = s + v[n];
}
double m(s / v.size());
```

Question 2 – Problème [63 points]

On s'intéresse ici à transmettre de façon compacte et sécurisée un son sur le réseau Internet. Plusieurs étapes sont donc nécessaires, parmi lesquelles : numériser le son, comprimer sa représentation, l'encrypter et transporter les paquets correspondant via IP.

Question 2.1 – Numérisation [8 points]

Question 2.1.1 – Échantillonnage [6 points]

Le son que l'on souhaite transmettre a été produit par un synthétiseur correspondant au code C++ ci-contre (dans lequel les temps sont en secondes et les fréquences en hertz).

Pour le numériser, on va prendre des échantillons toutes les 0.8 ms.

1. Peut-on reconstruire correctement le signal ainsi échantillonné en utilisant la formule de reconstruction vue en cours ?

2. Et si l'on échantillonnait toutes les 1 ms ?

Justifiez pleinement vos deux réponses et proposez une solution alternative dans chaque cas de réponse négative.

Réponse :

```
struct Composante {
    double amp;
    double freq;
    double phase;
};

const vector<Composante> signal({
    { 1.2,   333.5,  0.78  },
    { 0.78,  611.2,  0.5   },
    { 1.42,  440.0,  0.65  },
    { 0.33,  123.7,  0.44  },
    { 2.18,  432.5,  0.15  },
  });

double son(double t)
{
    double x(0.0);
    for (auto c : signal) {
        x += c.amp * sin( 2 * M_PI * c.freq * t
                          + c.phase );
    }
    return x;
}
```

suite au dos ☞

Question 2.1.2 – Quantification [2 points]

Sachant que l'amplitude de ce signal est toujours comprise entre -5.8 et 5.8, on décide de représenter les valeurs des échantillons en utilisant des nombres entiers signés sur 8 bits (à intervalles réguliers et en utilisant toutes les valeurs possibles). Quelle précision relative par rapport à l'amplitude maximale (que l'on supposera égale à 5.8 ici) a-t-on avec une telle représentation ? Donnez votre réponse sous forme de fraction et justifiez votre réponse.

Question 2.2 – Compression [27 points]

Question 2.2.1 – Calcul de l'entropie [13 points]

Après numérisation, on se retrouve donc avec une suite de valeurs que l'on peut représenter sur des `char` en C++ (8 bits).

Écrivez ici une *fonction* C++ pour calculer l'entropie (en bit) d'une telle suite de valeurs :

Question 2.2.2 – Taux de compression [2 points]

Votre programme précédent vous donne une entropie de 5.18 bit pour le signal considéré. Quel taux de compression pouvez vous alors espérer (bornes) ? Donnez votre réponse sous forme de deux fractions et justifiez brièvement votre réponse.

Question 2.2.3 – Compression [12 points]

Pour simplifier ici, supposons que l'on n'ait finalement que les sept valeurs suivantes : -127, -64, -32, 0, 32, 64, 127 observées respectivement 21, 432, 654, 1324, 444, 411 et 53 fois.

Proposez ici un code de compression optimal pour ces valeurs :

☞ En utilisant votre code, codez la séquence d'échantillons -64, 0, -32, 0, 32 :

suite au dos ☞

Question 2.3 – Sécurité [11 points]

Pour éviter tout vol de votre musique (droit d'auteur), vous souhaitez la transmettre en utilisant RSA.

Question 2.3.1 – Réencodage [2 points]

Pour cela, vous regroupez les mots de codes précédents pour en former à nouveau des entiers, en remplissant si nécessaire la fin par des 0.

Par exemple, si la séquence à transmettre est codée `10`, `0111`, `000`, `11`, `001`, vous regroupez les bits de sorte à transmettre `10011100`, `01100100`.

A quelle valeur décimale correspond le premier mot transmis ci-dessus (`10011100`), interprété comme entier non signé ?

Question 2.3.2 – Encryptage [3 points]

On interprète donc maintenant le message à transmettre comme une suite d'entiers positifs que l'on code chacun en utilisant RSA.

Votre clé publique est $(79, 899)$; votre clé privée est 319. La clé publique de votre destinataire est $(485, 667)$.

Comment transmettez vous l'entier 238 ? Donnez la réponse sous la forme « $a^b \mod c$ » :

Si vous recevez l'entier 532, en quoi le décodez vous ? Donnez la réponse sous la forme « $x^y \mod z$ » :

Question 2.3.3 – Encryptage en C++ [6 points]

Écrivez ici une fonction C++ capable de calculer la valeur (entier non signé) de $x^y \mod z$. Par exemple pour $x = 8$, $y = 27$ et $z = 55$, cette fonction retourne la valeur 2.

Question 2.4 – Transmission TCP/IP [17 points]

Il est maintenant temps de transmettre effectivement nos informations sur le réseau Internet (qui utilise le protocole TCP/IP).

Pour simplifier, nous ne mettrons qu'un seul entier par paquet réseau. Pour rappel, la couche TCP ajoute comme information à chaque paquet :

- une indication si c'est un paquet de donnée ou un paquet d'acquittement ;
- un numéro de séquence pour remettre les paquets dans l'ordre.

Question 2.4.1 – Paquet TCP [2 points]

Proposez ici un type C++ pour représenter les paquets TCP :

Question 2.4.2 – Ordre des paquets TCP [3 points]

En utilisant le type proposé ci-dessus, écrivez une fonction C++ qui, pour deux paquets donnés, indique s'ils sont dans le bon ordre ou non : elle retourne `true` si le paquet passé en premier argument arrive avant celui passé en second argument.

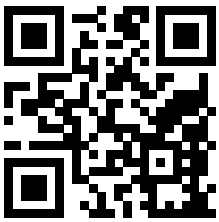
suite au dos ☞

Question 2.4.3 – Réception TCP [12 points]

Écrivez la fonction C++ qui gère la réception TCP des paquets : elle reçoit comme arguments un paquet P, un numéro i et une liste L de paquets ordonnés par numéro de séquence décroissant, et elle :

- ne fait rien et retourne i si P est un paquet d'acquittement ;
- crée le paquet d'acquittement correspondant à P (même contenu, seule l'information d'acquittement change) et l'envoie en utilisant une fonction `send()` que l'on suppose exister ;
- si le paquet n'est pas le suivant direct du dernier paquet reçu, dont le numéro est indiqué par l'argument i, elle ajoute P à sa place dans L et retourne i ; l'insertion de P à sa place dans L se fait en utilisant une fonction `insert()` que l'on suppose exister ;
- si le paquet P est le suivant direct du dernier paquet reçu (dont le numéro est indiqué par l'argument i), elle l'affiche en utilisant une fonction `affiche_paquet()` (au singulier), que l'on suppose exister ;
  si, de plus, le dernier paquet de L est celui attendu juste après P, elle affiche également tous les paquets jusqu'au prochain paquet manquant et les supprime de L (exemple ci-dessous) ;
  dans ce cas, la fonction retourne le numéro de séquence du dernier paquet affiché ;

  par exemple, si i vaut 42, et que le numéro de séquence de P est 43 et que L contient, dans cet ordre, des paquets dont les numéros de séquence sont 55, 50, 46, 45, 44, cette fonction affichera alors P et les paquets 44 à 46 (dans cet ordre) ; elle retournera 46 et la liste L ne contiendra alors plus que les paquets 55 et 50 (dans cet ordre).



suite au dos ☞

Question 3 – Mélange de mots [12 points]

Question 3.1 [3 points]

Quelle est l'entropie (telle que définie en cours) d'un mot X constitué de n fois la lettre `'F'` et m fois la lettre `'G'` : « $\overbrace{\texttt{F}\ldots\ldots}^{n}\overbrace{\texttt{FG}\ldots\ldots}^{m}\texttt{G}$ » ? Soit $p = \dfrac{n}{n+m}$. Utilisez p pour exprimer cette entropie comme une fonction de p uniquement :

$H(X) = h(p) =$

Indication : que vaut $1 - p$?

Justifiez votre réponse simplement en explicitant le calcul effectué.

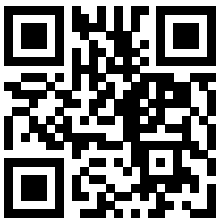
Question 3.2 [9 points]

Soient A un mot de longueur n et B un mot de longueur m n'ayant *aucune* lettre en commun avec A. Démontrez que l'entropie (telle que définie en cours) du mot « AB » , constitué du mot B collé derrière le mot A vaut

$$H(AB) = h(p) + p\,H(A) + (1-p)\,H(B)$$

où $p = \dfrac{n}{n+m}$ et $H(A)$, $H(B)$ sont respectivement l'entropie du mot A et du mot B (telle que définie en cours).

Indication : imaginez le calcul de $H(A)$, celui de $H(B)$ et faites le calcul de $H(AB)$.



suite au dos ☞

Question 4 – Compréhension de programme [16 points]

L'extrait de programme suivant ne contient pas d'erreur :

```
bool f(double a, vector<double> const& v, size_t i = 0, double prec = 1e-10)
{
    return (i < v.size()) and
           ( (abs(a - v[i]) < prec)  or  f(a, v, i+1, prec) );
}

vector<double> g(vector<double> const& v1, size_t i, vector<double> const& v2)
{
    vector<double> res;
    if (i == 0) return res;
    --i;
    res = g(v1, i, v2);
    if ((i < v1.size()) and not f(v1[i], v2)) {
        res.push_back(v1[i]);
    }
    return res;
}

vector<double> g(vector<double> const& v1, vector<double> const& v2)
{
    return g(v1, v1.size(), v2);
}
```

Question 4.1 [6 points]

Expliquer *brièvement* ce que font chacune des fonctions `f` et `g` et proposer un nom plus pertinent pour chacune d'entre elles.

Question 4.2 [10 points]

Si la complexité de `size()` et de `push_back()` pour un `vector` sont en $\mathcal{O}(1)$, quelle est la complexité de l'algorithme équivalent à la fonction `g` en fonction de la taille de `v1` ? en fonction de la taille de `v2` ? Justifiez vos réponses.

suite au dos ☞

Question 5 – La cache est déterminante [19 points]

On considère l'extrait de programme suivant (que l'on suppose correct) :

```
double det2(double a, double b, double c, double d) {
    return a*d - b*c;
}
double det3(array<array<double, 3>, 3> const& mat) {
    return det2(mat[1][0], mat[1][1], mat[2][0], mat[2][1]) * mat[0][2]
         - det2(mat[2][1], mat[2][0], mat[0][1], mat[0][0]) * mat[1][2]
         + det2(mat[0][0], mat[0][1], mat[1][0], mat[1][1]) * mat[2][2];
}
```

On suppose que les `double` utilisent 2 mots mémoire, que la cache a deux blocs de 4 mots chacun et que `matrice` est un tableau 3x3 stocké par lignes : si la valeur (en mot) de `&(matrice[0][0])` est a, alors la valeur (en mot) de `&(matrice[0][1])` est $a+2$ et `&(matrice[1][0])` est $a+6$.

Le but de cette question est de calculer le nombre de défauts de cache lors du calcul de `det3(matrice)`.

Question 5.1 Schéma mémoire [7 points]

Dessiner la cache (sans valeur) et les éléments en mémoire avec la convention qu'un rectangle correspond à un mot. Écrivez simplement M_{10} pour « `matrice[1][0]` », etc. Donnez également les adresses (p.ex. « $a+6$ » au dessus de quelques cases pertinentes). La ligne « accès : » au dessous de la case n'est utile que pour la question suivante.

Cache :

Mémoire (off-chip) :

adresse :

contenu : | | | | | | | | | | |

accès :

adresse :

contenu : | | | | | | | | | | |

accès :

Question 5.2 Ordre d'appel [4 points]

En supposant que les expressions sont évaluées dans l'*ordre de lecture* du code C++ précédent, indiquez ci-dessus (dans la ligne « accès ») l'ordre d'accès (de 1 à 15) aux différents éléments mémoire. Mettez simplement le ou les numéros d'ordre d'accès sous chaque case mémoire correspondante.

Par exemple, si la donnée que vous avez mise dans la 3e case ci-dessus est la 5e et la 8e à être accédée lors du calcul de l'appel `det3(matrice)`, mettez simplement « 5, 8 » sous la 3e case ci-dessous.

Question 5.3 Nombre de défauts de cache [8 points]

En supposant que l'adresse $a =$`&(matrice[0][0])` est alignée sur les blocs de la cache,

1. marquez (clairement) les accès mémoire qui provoquent un défaut de cache parmi les 15 accès mémoire effectués pour l'appel `det3(matrice)` :

   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

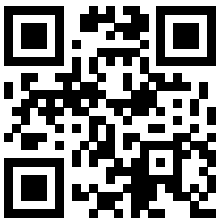
   expliquez/justifiez votre réponse ;
2. concluez en donnant le nombre total de défauts de cache.

suite au dos ☞

Question 6 – Chercher les erreurs [14 points]

Considérer le programme C++ suivant (sur deux pages) :

```
1   #include <iostream>
2   #include <string>
3   #include <vector>
4   #include <array>
5   #include <fstream>
6   using namespace std;
7
8   typedef vector<string> Texte;
9
10  // ============================================================================
11  string encodage(Texte const& en_clair)
12  {
13    const array<string, 16> code({
14        "01",   "110",   "101",   "1111",
15        "0100", "1010",  "00101", "1110",
16        "00",   "0011",  "11001", "111000",
17        "0101", "11"  ,  "01011", "101010"
18      });
19
20    string resultat;
21    for (auto mot : en_clair) {
22      for (auto c : mot) {
23        while (c > 16) {
24          resultat += code[c % 16];
25          c -= 16;
26        }
27        resultat += code[c];
28      }
29    }
30    return resultat;
31  }
32
33  // ============================================================================
34  Texte lire_fichier(string const& nom)
35  {
36    Texte retour;
37    ifstream fichier(nom);
38    string mot;
39    size_t i(0);
40    while (fichier << mot) {
41      retour[i] = mot;
42      i += 1;
43    }
44    return retour;
45  }
46
```



```
47  // ========================================================================
48  string demander_nom_fichier()
49  {
50    string retour("test.txt");
51    cout << "Entrez un nom de fichier : ";
52    getline(cin, retour);
53    return retour;
54  }
55
56  // ========================================================================
57  int main()
58  {
59    cout << encodage(lire_fichier(demander_nom_fichier()))
60         << endl;
61    return 0;
62  }
```

Question 6.1 – But général [2 points]

Bien que ce programme contienne plusieurs erreurs, indiquer (*brièvement* mais *clairement*) le but et le fonctionnement général attendu de ce programme (c.-à-d. l'intention du programmeur).

Question 6.2 – Compilation [3 points]

Ce programme ne compile pas correctement. Le compilateur n'indique qu'une seule erreur :

```
pgm.cc:40:18: error: no match for ‘operator<<’ (operand types are ‘std::ifstream’
              {aka ‘std::basic_ifstream<char>’} and ‘std::__cxx11::string’
              {aka ‘std::__cxx11::basic_string<char>’})
```

Que cela signifie-t-il ? (quelle est l'erreur ?) Comment corriger ?
Répondez sur le code lui-même en marquant votre reponse d'un « Question 6.2 ».

Question 6.3 – Analyse critique [9 points]

Ce programme comporte encore au moins quatre autres erreurs de différente nature (théorie, conception, implémentation, mauvaises pratiques, etc.)

Indiquez ces erreurs puis corrigez les.
Vous pouvez répondre sur le code lui-même (sans autre marque) ou sur la page au dos.

Attention ! Il y aura des pénalités pour des indications d'erreur qui n'en sont pas vraiment (mais seront tolérents sur les mauvaises pratiques).