ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

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

Information, Calcul et Communication (SPH) : Correction de l'examen final

19 décembre 2025

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 feuille supplémentaire ; **seul ce document sera corrigé**.

   Si nécessaire, il y a une page de réponse supplémentaire en fin de copie (page 19).

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(e) des assistant(e)s.

6. L'examen comporte 5 exercices indépendants sur 19 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 125 points).

   Tous les exercices comptent pour la note finale.

Question 1 – Échauffement [9 points]

① **[5 points]** En utilisant RSA, un ami, dont la clé publique est $(2999, 4891)$, vous a transmis le message confidentiel 2954.

Vous avez de votre coté égaré vos clés, mais aviez noté p, q, e et d sur un papier ; vous ne savez cependant plus dans quel ordre. Voici ce que vous retrouvez sur votre papier :

$$995 \qquad 43 \qquad 155 \qquad 73$$

Par contre, vous vous souvenez que vous aviez publié 155.

Comment décodez vous le message reçu de votre ami ?
Exprimez votre réponse sous la forme « $x^y \mod z$ » et **justifiez** pleinement votre réponse.

Réponse et justification : Pour décoder le message reçu $M = 2954$, il faut faire $M^d \mod n$ avec notre d (clé privée) et notre n.

Il nous faut donc retrouver d et $n = p\,q$ parmi $995, 43, 155$ et 73. On sait qu'on a publié 155, qui est donc public ; c'est donc soit n, soit e (tout le reste est privé).

Or (deux raisonnements possibles) :

- 155 est dans la liste donnée ci-dessus (p, q, e et d) et il n'est pas possible d'avoir $e = n$;
- 155 est clairement divisible par 5 (qui serait p ou q si 155 était n), lequel n'est pas dans notre liste de nombres.

Donc 155 est nécessairement notre e.

Parmi $995, 43$ et 73, 995 est aussi divisible par 5 et donc n'est pas premier (donc ni p ni q).

On en déduit que pour nous (p et q sont bien sûr interchangeables) :

$$d = 995 \qquad p = 43 \qquad e = 155 \qquad q = 73$$

et donc

$$n = 43 \times 73 \big(= 3139, \text{ non demandé}\big)$$

Finalement, on décode donc comme :

$$2954^{995} \mod (43 \times 73)$$

Commentaire : Plusieurs confusions entre les différents éléments de RSA. Même quelques confusions sur quel n (et d ou e) utiliser dans ce contexte ci.

② **[4 points]** On considère le signal suivant :

$$X(t) = 2\,\sin\left(7\pi\,t + \frac{3\pi}{2}\right) + 3\,\sin\left(29\pi\,t + \frac{\pi}{3}\right) + 4\,\sin\left(15\pi\,t + \frac{5\pi}{6}\right) + 5\,\sin\left(20\pi\,t + \frac{3\pi}{4}\right)$$

lequel est tout d'abord filtré avec un filtre passe-bas idéal de fréquence de coupure 14 Hz avant d'être échantillonné.

À quelle fréquence au minimum faut-il échantillonner le signal filtré pour pouvoir le reconstruire parfaitement (avec la formule de reconstruction vue en cours) ?

Justifiez pleinement votre réponse.

Réponse et justification : Le signal après filtrage à $f_c = 14$ Hz est (pas explicitement demandé) :

$$\widehat{X}(t) = 2\,\sin\left(7\pi\,t + \frac{3\pi}{2}\right) + 4\,\sin\left(15\pi\,t + \frac{5\pi}{6}\right) + 5\,\sin\left(20\pi\,t + \frac{3\pi}{4}\right)$$

dont la bande passante est $f_{\max} = 10$ Hz.

Il faut donc échantillonner à au moins (strictement) 2×10 Hz, p.ex. à $f_e = 21$ Hz.

Commentaire : Attention à l'inégalité **stricte** dans l'application du théorème de Nyquist-Shannon. Aussi quelques manques de justification.

Question 2 – Tout récursif [32 points]

① **[5 points]** Écrivez un algorithme **somme**, *récursif* et sans boucle qui prend en entrée une liste de nombres entiers et sort la somme de tous ces nombres.

Réponse :

Voici une version possible :

somme
entrée : *L liste de nombres entiers* sortie : *somme de ses éléments*
$n \longleftarrow$ **taille**(L) **Si** $n = 0$ **Sortir :** 0 **Sortir :** $L[1] +$ **somme** $(L[2], ..., L[n])$

(sous-entendu que $(L[2], ..., L[n])$ est la liste vide si $n = 1$; et sinon, distinguez explicitement le cas $n = 1$: **Sortir :** $L[1]$.)

Commentaire : (valable ici ou pour les algorithmes suivants) Encore beaucoup trop oublient de traiter le cas de la liste vide.
Quelques un(e)s sortent des listes au lieu d'une valeur, en particulier la liste vide au lieu de 0.
Et encore trop d'oublis de l'utilisation de la valeur laissée de coté lors de l'appel récursif (en clair : oubli d'additionner $L[1]$).

② **[4 points]** Quelle est la complexité de votre algorithme **somme** écrit en ① ? **Justifiez** votre réponse.

Réponse : La complexité dépend bien sûr de l'algorithme proposé.

Pour l'algorithme ci-dessus, mis à part l'appel récursif, tout le reste sont des opérations élémentaires (on peut supposer que `taille` est en $\Theta(1)$). La complexité vient donc du nombre d'appels, qui ici est linéaire puisque l'on enlève un élément à chaque fois.

Commentaire : Il ne faut pas oublier de justifier que « tout le reste sont des opérations élémentaires » (sinon la conclusion serait différente).

③ **[7 points]** On souhaite maintenant utiliser l'algorithme **somme** précédent et un algorithme **Hrec** à définir pour pouvoir calculer l'entropie d'une liste L de comptes de lettres, $L = (n_1, ..., n_n)$, de la façon suivante :

entropie
entrée : *liste L telle que décrite ci-dessus* sortie : *entropie des comptes fournis*
$n \longleftarrow$ **taille**(L) $S \longleftarrow$ **somme**(L) **Sortir : Hrec**(n, S, L)

Écrivez l'algorithme **Hrec** de façon *récursive* et sans boucle.

Réponse :

C'est la même question que ① mais appliquée à la liste des $n_i/S \log(S/n_i)$. Il n'est cependant pas question ici de construire cette liste avec une boucle.

Attention aussi au cas des n_i nuls (ne pas diviser par 0, ou prendre le log de 0).
Note : N ne peut être nul que si tous les n_i sont nuls.

Voici une version élégante (sans réduire L) :

Hrec

entrée : n, S *et* L *tels que décrits ci-dessus*
sortie : *entropie des comptes dans* L

Si $n \leq 1$
  Sortir : 0
Si $L[n] = 0$
  Sortir : Hrec $(n-1, S, L)$
Sortir : $L[n]/S \log(S/L[n])$ + **Hrec** $(n-1, S, L)$

et une version plus explicite :

Hrec

entrée : n, S *et* L *tels que décrits ci-dessus*
sortie : *entropie des comptes dans* L

Si $n = 0$
  Sortir : 0
Si $L[1] = 0$
  Sortir : Hrec $(n-1, S, L[2], ..., L[n])$
Sortir : $L[1]/S \log(S/L[1])$ + **Hrec** $(n-1, S, L[2], ..., L[n])$

Commentaire : Très peu pensent à traiter le cas $L[1] = 0$.

④ **[2 points]** Quelle est la complexité de votre algorithme **Hrec** écrit en ③ ? **Justifiez** votre réponse.

Réponse : Exactement comme pour ②.

⑤ **[10 points]** Écrivez un algorithme **est_trié**, *récursif* et sans boucle qui prend en entrée une liste de nombres entiers et sort « vrai » si la liste est triée (par ordre croissant ou décroissant, peu importe ; les deux cas doivent fonctionner) et « faux » si la liste n'est pas triée.

Remarque : testez peut-être votre algorithme avec $(4, 5, 5, 4, 3, 2)$, $(2, 3, 4, 5, 5, 4)$ ou $(2, 3, 4, 4, 4, 3, 2)$.

Réponse :

Voici une version possible (écrite de façon « éclatée », mais on peut bien sûr adopter une écriture plus compacte avec une seule ligne « **Sortir** ») :

est_trié
entrée : *liste L telle que décrite*
sortie : *triée ou pas ?*

$n \longleftarrow$ **taille**(L)
Si $n \leq 2$
  Sortir : vrai
$a \longleftarrow$ **est_trié**$\big(L[2], \cdots, L[n]\big)$
$b \longleftarrow (L[1] \leq L[2])$ **et** $(L[2] \leq L[n])$
$c \longleftarrow (L[1] \geq L[2])$ **et** $(L[2] \geq L[n])$
Sortir : a **et** $(b$ **ou** $c)$

Note : il y a plein de façons d'écrire $(b$ **ou** $c)$; en particulier $(b$ **ou** $c)$ est vrai si $L[1] = L[2]$ ou si $L[2] = L[n]$, et donc, si ces cas ont été testés par ailleurs, on peut aussi avoir des inégalités strictes.

Voici une autre solution correcte qui part sur une autre voie :

est_trié
entrée : *liste L telle que décrite*
sortie : *triée ou pas ?*

Sortir : est_ordonnee$(L, \leq)$ **ou est_ordonnee**$(L, \geq)$

avec :

est_ordonnee
entrée : *liste L telle que décrite, $\perp$ une relation d'ordre*
sortie : *L est ordonnée par $\perp$ ou pas ?*

$n \longleftarrow$ **taille**(L)
Si $n \leq 1$
  Sortir : vrai
Sortir : $\big(L[1] \perp L[2]\big)$ **et est_ordonnee**$\big((L[2], ..., L[n]), \perp\big)$

Attention aux versions trop simplistes qui ne fonctionnent pas sur les exemples donnés comme p.ex. :

est_trié

entrée : *liste L telle que décrite*
sortie : *triée ou pas ?*

$n \longleftarrow$ **taille**(L)
Si $n \leq 2$
  Sortir : vrai
$a \longleftarrow$ **est_trié**$\big(L[2], \cdots, L[n]\big)$
$b \longleftarrow (L[1] \leq L[2])$ **et** $(L[2] \leq L[3])$
$c \longleftarrow (L[1] \geq L[2])$ **et** $(L[2] \geq L[3])$
Sortir : a **et** $(b$ **ou** $c)$

ou :

est_trié

entrée : *liste L telle que décrite*
sortie : *triée ou pas ?*

$n \longleftarrow$ **taille**(L)
Si $n \leq 2$
  Sortir : vrai
$a \longleftarrow$ **est_trié**$\big(L[1], \cdots, L[\lfloor \frac{n-1}{2} \rfloor]\big)$
$d \longleftarrow$ **est_trié**$\big(L[\lfloor \frac{n+3}{2} \rfloor], \cdots, L[n]\big)$
$b \longleftarrow (L[\lfloor \frac{n-1}{2} \rfloor] \leq L[\lfloor \frac{n+1}{2} \rfloor])$ **et** $(L[\lfloor \frac{n+1}{2} \rfloor] \leq L[\lfloor \frac{n+3}{2} \rfloor])$
$c \longleftarrow (L[\lfloor \frac{n-1}{2} \rfloor] \geq L[\lfloor \frac{n+1}{2} \rfloor])$ **et** $(L[\lfloor \frac{n+1}{2} \rfloor] \geq L[\lfloor \frac{n+3}{2} \rfloor])$
Sortir : a **et** d **et** $(b$ **ou** $c)$

Pour la version « coupé en deux », b et c doivent en fait être plus compliqués ; p.ex. pour b :

$(L[1] \leq L[\lfloor \frac{n-1}{2} \rfloor])$ **et** $(L[\lfloor \frac{n-1}{2} \rfloor] \leq L[\lfloor \frac{n+1}{2} \rfloor])$ **et** $(L[\lfloor \frac{n+1}{2} \rfloor] \leq L[\lfloor \frac{n+3}{2} \rfloor])$ **et** $(L[\lfloor \frac{n+3}{2} \rfloor] \leq L[n])$

⑥ **[4 points]** Quelle est la complexité de votre algorithme **est_trié** écrit en ⑤ ? **Justifiez** votre réponse.

Réponse : La complexité dépend bien sûr de l'algorithme proposé.

Les premiers algorithmes proposés sont en $\Theta(n)$, n étant la taille de la liste en entrée : on retire un élément à chaque et, mis à part l'appel récursif, tout le reste sont des opérations élémentaires (on peut supposer que `taille` est en $\Theta(1)$).

Le dernier algorithme proposé (« couper en deux ») est aussi linéaire. Même si c'est l'adaptation à ce problème de l'idée de l'algorithme de recherche par dichotomie, on passe ici par *tous* les sous-appels.

La justification de cette complexité peut se faire par l'arbre des appels ($\log_2(n)$ étages, complexité $\Theta(1)$ à *chaque* étage de l'arbre, **mais** on passe par tous les nœuds de l'arbre, donc $2^{1+\log_2 n} - 1$ au total), ou formellement :

$$C(n) = a + 2\,C(n/2)$$

avec $C()$ la complexité de l'algorithme proposé.

Commentaire : Beaucoup de confusion ici. Peu arrivent à clairement séparer les deux cas (seconde solution ci-dessus) et s'embrouillent dans les cas croissants puis décroissants (ou symétriques ; cas d'erreurs mentionnés dans le corrigé ci-dessus).

Question 3 – Réseau [23 points]

Note : (comme toujours, mais particulièrement ici) nous vous conseillons, afin de faire les bons choix, de lire *entièrement* cette question avant de commencer.

Nous souhaitons écrire (des parties d')un programme C++ de simulation de réseau TCP, et plus précisément les nœuds de ce réseau et leurs connexions.

Pour cela, nous avons déjà à disposition les définitions suivantes des types permettant de représenter des adresses IP (type `AdresseIP`) et des tables de routage (type `Table`) :

```
typedef string AdresseIP;

struct Ligne {
  AdresseIP destination;
  AdresseIP proche_voisin;
  unsigned int distance;
};

typedef vector<Ligne> Table;
```

① **[3 points]** En utilisant les types définis ci-dessus, on vous demande de définir ici le type `Noeud` pour représenter un nœud du réseau. Un tel nœud doit contenir :

- son adresse IP ;
- sa table de routage ;
- la liste de ses voisins (*libre* à vous de choisir comment le faire) ;
- un booléen `deja_vu` indiquant si le nœud a déjà été visité ou non ; celui-ci nous servira lors de parcours du réseau (voir la sous-question ② suivante).

Réponse :

```
struct Noeud {
  AdresseIP adresse;
  Table table;
  vector<Noeud*> voisins; // peuvent aussi utiliser set<> s'ils connaissent
  bool deja_vu;
};
```

Les voisins peuvent difficilement être autre chose qu'un pointeur, ne serait-ce que pour faire l'affichage *récursif* correctement.

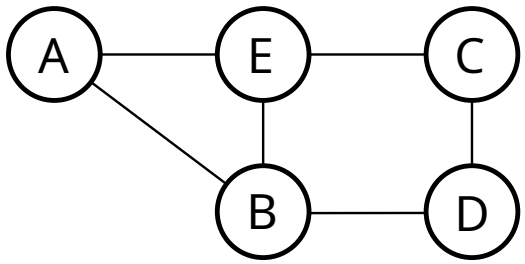
Commentaire : Peu d'élèves pensent à faire de `voisins` un tableau de *pointeurs* sur des nœuds : beaucoup ont fait des tableaux d'`AdresseIP`, et quelques-uns ont voulu faire des `vectors` de références.

Et beaucoup trop écrivent des variations du style « `vector<Ligne.some_field> voisins` ».

② **[9 points]** Définissez ci-dessous la fonction `affiche()` qui prend un `Noeud` et une chaîne de caractères et qui affiche (récursivement) tout le réseau à partir de ce nœud.

Par exemple, pour le réseau ci-dessous à droite, l'appel à cette fonction depuis le nœud A donnera :

```
- A connecté à :
    - B connecté à :
        - A
        - D connecté à :
            - B
            - C connecté à :
                - D
                - E connecté à :
                    - A
                    - B
                    - C
        - E
    - E
```



La chaîne de caractères passé en second argument sert à décaler l'affichage de chaque voisin : elle doit avoir une valeur par défaut de `"- "` et sera augmentée devant de `"    "` (4 espaces) pour chaque nouveau niveau de voisins.

Notez bien que les voisins d'un nœud ne sont affichés que la première fois que ce nœud est affiché (ils ne sont plus affichés si le nœud a déjà été affiché ; utilisez le champ `deja_vu` des nœuds pour cela).

Réponse :

```
void affiche(Noeud& n, string offset = "- ")
{
  n.deja_vu = true;
  cout << offset << n.adresse << " connecté à :" << endl;
  for (auto v : n.voisins) {
    if (not v->deja_vu) {
      affiche(*v, "    " + offset);
    } else {
      cout << ("    " + offset) << v->adresse << endl;
    }
  }
}
```

Commentaire : Les trois erreurs principales étaient de

1. oublier le nœud de départ
2. passer le nœud par référence constante (ou par copie)
3. et de ne pas prendre en compte le fait qu'un nœud ait déjà été affiché (donc ne pas se servir de `deja_vu`).

   (Il est alors nécessaire de passer le nœud par référence (non constante).)

Aussi beaucoup de problèmes avec l'utilisation des pointeurs (p.ex. `*ptr.addresse` ou, question suivante, `*voisin == node2`)

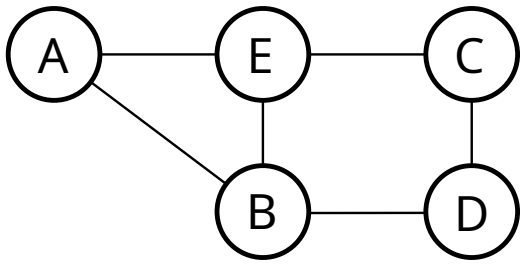
③ **[11 points]** On vous demande enfin d'écrire la fonction `connecte()` qui prend deux nœuds en argument et qui, *si* les nœuds ne sont pas déjà voisins :

— les connecte (ajoute *chacun* aux voisins de l'autre) ;

— met à jour leur table de routage pour ce nouveau nœud ajouté
(s'il existe déjà dans la table, mettre sa distance à 1 et sinon l'ajouter à distance 1).

Par exemple, pour représenter le réseau ci-contre,
on aurait écrit :

```
connecte(A, B);
connecte(A, E);
connecte(B, D);
connecte(B, E);
connecte(C, D);
connecte(C, E);
```



Réponse : Voici une réponse possible, bien modularisée :

```
bool est_connecte(Noeud const& A, Noeud const& B)
{
  for (auto v : A.voisins) if (v == &B) return true;
  return false;
}

void mise_a_jour(Table& table, Noeud const& B)
{
  for (auto& ligne : table) {
    if (ligne.destination == B.adresse) {
      ligne.proche_voisin = B.adresse;
      ligne.distance = 1;
      return;
    }
  }

  // pas trouvé
  table.push_back({ B.adresse, B.adresse, 1 });
}

void connecte(Noeud& A, Noeud& B)
{
  if (not est_connecte(A, B)) { // set semantics
    A.voisins.push_back(&B);
    B.voisins.push_back(&A);

    mise_a_jour(A.table, B);
    mise_a_jour(B.table, A);
  }
}
```

En voici une autre possible :

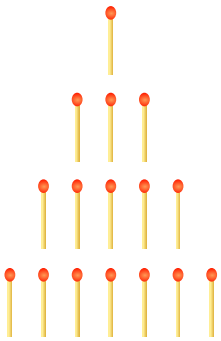
```
void connecte_deux_a_un(Noeud& A, Noeud& B)
{
  for (auto v : A.voisins) if (v == &B) return; // déjà voisins

  A.voisins.push_back(&B);

  for (auto& ligne : A.table) {
    if (ligne.destination == B.adresse) {
      ligne.proche_voisin = B.adresse;
      ligne.distance = 1;
      return;
    }
  }

  // pas trouvé
  A.table.push_back({ B.adresse, B.adresse, 1 });
}

void connecte(Noeud& A, Noeud& B)
{
  connecte_deux_a_un(A, B);
  connecte_deux_a_un(B, A);
}
```

Commentaire : Parmi les erreurs courantes, il y avait le fait de ne pas mettre à jour `proche_voisin`, ainsi que la confusion entre « être voisin » et « être dans la table de l'autre ».

Souvent, la mise à jour de la table de routage est faite uniquement dans le cas où les deux nœuds sont déjà voisins.

Aussi, très peu ont écrit de fonctions intermédiaires et il y avait (trop) souvent des « copiés-collés ».

Question 4 – Retour à Marienbad [29 points]



© M0tty, CC BY-SA 3.0, via Wikimedia Commons

On s'intéresse ici des parties d'un programme permettant de jouer au jeu de Marienbad, lequel est constitué de quatre lignes ayant respectivement 1, 3, 5 et 7 allumettes (cf image ci-contre).

Les premières questions s'intéressent au codage des situations du jeu, les dernières questions à la sauvegarde sur fichier, en C++, et à un bug possible.

① **[2.5 points]** Pour coder une situation du jeu, si l'on code simplement le nombre d'allumettes de chaque ligne séparément, de combien de bits a-t-on besoin en tout pour coder ?

Justifiez votre réponse.

Réponse et justification : 2 situations (0 ou 1) nécessitent 1 bit ; 4 situations, 2 bits ; 6, 3 bits et 8, 3 bits aussi, et donc au total $1+2+3+3=9$ bits.

Commentaire : Manque de justifications et non prise en compte la situation avec 0 allumettes.
On a même vu plusieurs copies qui affirment sans gène que $\log_2(1)=1$!

② **[4 points]** En utilisant le codage de la question ① précédente, avec en premier (bit(s) de poids fort(s)) le ou les bit(s) représentant la première ligne (0 ou 1 allumette), puis juste après le ou les bits représentant la seconde ligne (0 à 3 allumette(s)), etc., quelle serait la valeur équivalente en décimal (convention non signée) correspondant à la représentation de la situation de jeu 1,2,3,5 ?

Justifiez votre réponse.

Réponse et justification : $256+2\times64+3\times8+5=413$

Commentaire : Plusieurs ne donnent pas la valeur décimale comme demandé.

③ **[2 points]** En utilisant le codage de la question ② précédente, quelle est la situation de jeu correspondant à la valeur décimale 66 ?

Justifiez votre réponse.

Réponse et justification : 0,1,0,2 : rien sur 256 (car plus petit), 1 sur 64, puis le reste.

④ **[1 point]** Il y a en fait $2\times4\times6\times8=384$ situations de jeu possibles. En utilisant un même nombre de bits pour coder chacune de ces situations, de combien de bits a-t-on besoin au minimum ?

Justifiez votre réponse.

Réponse et justification :

$$\lceil \log_2 384 \rceil = 9 \qquad (256 < 384 < 512)$$

Commentaire : Certain(e)s ont compris la question comme « *Combien faudrait-il de bits pour représenter les 384 états tous ensembles ?* » (comme un tableau qui contiendrait la représentation de chacun des états). Pourquoi voudrait-on faire ça (`tab[i]=i`) ?

⑤ **[4.5 points]** Combien de valeurs différentes peut-on coder sur le nombre de bits que vous avez indiqué à la question ④ précédente ?
Combien de valeurs différentes peut-on coder sur le nombre de bits que vous avez utilisé à la question ① ?
Expliquez pourquoi il y a un écart entre ce nombre de valeurs et 384 ; p.ex., quelle valeur ne correspond pas à une situation de jeu ?
Retrouver le nombre 384 en comptant le nombre de valeurs qui ne correspondent pas à des situations de jeu dans le codage proposé en questions ① à ③.

Réponses et justifications : C'est le même nombre de bits que pour la question ① (qui donc, ne perd rien). Il permet de représenter 512 valeurs différentes.

L'écart provient des 3 bits utilisé pour coder les 6 valeurs de 0 à 5. On pourrait aussi y coder 6 et 7 mais elles ne correspondent pas à des valeurs possibles pour le nombre d'allumettes en troisième ligne.

Par exemple la valeur $48 = 6 \times 8$ ne correspond à rien (ce serait 0,0,6,0).

Il y a 64 situations différentes possibles pour les autres lignes que la troisième ($2 \times 4 \times 8 = 64$). On a donc $64 \times (2^3 - 6) = 128$ valeurs non utilisées (qui ne correspondent pas à des situations de jeu) et 384 vaut bien $512 - 128$.

Commentaire : Certain(e)s étudiant(e)s ont répondu que le nombre de valeurs représentables est 511 au lieu des 512.
Plusieurs ne justifient pas assez (p.ex. *combien* de situations manquent).

⑥ **[10.5 points]** Le programme C++ que l'on a écrit nécessite de mémoriser des valeurs dans un tableau `Memory` :

```
typedef array<int, 496> Memory;
```

que l'on souhaite pouvoir lire depuis un fichier et aussi sauvegarder dans un fichier.

Écrivez ci-contre deux fonctions `lire()` et `sauvegarder()` qui prennent une `Memory` et un nom de fichier en argument et ne retournent rien.
Ces fonctions devront lancer une/des exception(s) de votre choix en cas d'erreur.

Réponse : Voici une version minimale :

```
void sauvegarder(Memory const& mem, string const& nom)
{
  ofstream output(nom);
  if (output.fail()) throw 42;
  for (auto score : mem) output << score << " ";
  output << endl; // optionnel
  output.close(); // optionnel
}

void lire(Memory& mem, string const& nom)
{
  ifstream input(nom);
  if (input.fail()) throw 42;
  for (auto& score : mem) input >> score;
  input.close(); // optionnel
}
```

Ce qu'il ne faut pas rater, ce sont le passage par référence et le parcours par référence dans la fonction `lire()`.

On peut aussi bien sûr :
- sophistiquer les exceptions ;
- tester le succès des écritures/lectures (et lancer des exceptions sinon) ;
- utiliser des `string_view` ;
- écrire (et lire) les `Memory` en binaire :

(voir page suivante)

```
void sauvegarder(Memory const& mem, string const& nom)
{
  ofstream output(nom, ios::binary);
  if (output.fail()) throw 42;
  output.write(reinterpret_cast<const char*>(mem.data()), sizeof mem);
}

void lire(Memory& mem, string const& nom)
{
  ifstream input(nom, ios::binary);
  if (input.fail()) throw 42;
  input.read(reinterpret_cast<char*>(mem.data()), sizeof mem);
}
```

Commentaire : La programmation est certainement la partie la moins réussie, avec comme fautes plus courantes :

- d'oublier le passage par référence de la mémoire,
- de faire des `cerr << ...` à la place d'un `throw`,
- de mettre des blocs `try catch` dans les fonctions,
- de prendre un `stream` en argument à la place d'un `string`
- ou tout simplement d'oublier de lire/écrire dans le fichier après l'avoir ouvert et testé son ouverture.
- Et encore des `int` à la place de `size_t`...

⑦ **[4.5 points]** Voici un extrait du code produit, lequel est censé faire apprendre à la machine comment mieux jouer :

```
void learn(Memory mem, Play_record record, bool won = true)
{
  Score incr(1);
  if (won) incr = -1;
  for (auto game_rep : record) {
    mem[game_rep] += incr;
    incr = -incr;
  }

  // if last move results from a resign: don't take it into account
  if (record.back() != 0) mem[record.back()] += incr;
}
```

Et voici un extrait de comment cette fonction `learn()` est appelée :

```
int main()
{
  Memory ia_memory{};
  Play_record record;
  bool ia_wins(true);

  //... plein d'autres choses

  learn(ia_memory, record, ia_wins);

  return 0;
}
```

Mais après plusieurs essais, le programme refuse d'apprendre.

Corriger le code ci-dessus pour le programme puisse effectivement apprendre.
Commentez brièvement votre correction ou la/les source(s) d'erreur.

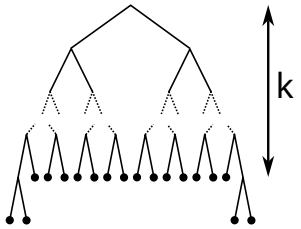
L'erreur est simplement l'oubli du passage par référence de `mem`.

Commentaire : Moins de la majorité des élèves ont trouvé cette erreur, mais beaucoup ont fait preuve d'une créativité surprenante. De manière aussi surprenante, une part non négligeable de celles et ceux qui ont trouvé l'erreur ont eux-/elles-mêmes oublié le passage par référence de la mémoire dans leur code à la sous-question précédente.

Question 5 – Huffman et les autres [32 points]

On s'intéresse ici à coder (en binaire) des séquences faites à partir de n lettres différentes (toutes présentes), avec n de la forme $n = 2^k + 2$ pour $k \geq 2$. Par exemple pour $k = 3$ ($n = 10$), on voudrait coder la séquence Y = ABCDE FGHIJ BBA HHIIII CAGE CFF (sans les blancs).

① **[5.5 points]** On considère ici le code binaire $\mathcal{C}$ dont l'arbre de codage est constitué (cf image ci-contre) d'un arbre binaire de profondeur k dont deux feuilles ont été chacune remplacée par un sous-arbre binaire de profondeur 1.



- **(a)** Est-il toujours possible de construire un tel code (sous les contraintes données plus haut) ?
- **(b)** Si l'on note p_1, p_2, p_3 et p_4 les quatre probabilités des lettres les plus en bas de l'arbre de codage, quelle est la longueur moyenne d'un tel code ?
- **(c)** Un tel arbre de codage pourrait-il être un arbre de code de Huffman ? Si oui, à quelle condition (suffisante) ?

Justifiez *chacune* de vos réponses.

Réponses et justifications :

- **(a)** Oui parce qu'il y a 2^k feuilles dans l'arbre binaire de profondeur k, donc $2^k - 2$ feuilles de profondeur k dans l'arbre de $\mathcal{C}$ (note : $k \geq 2$ donc $2^k - 2 \geq 2$), et donc les deux places restantes pour les deux sous-arbres de profondeur 1.
  Ce qui fait au total $2^k - 2 + 4 = 2^k + 2$ feuilles. On a donc bien la place pour coder toutes les n lettres (et exactement elles).
- **(b)** La somme des probabilités des lettres de code de profondeur k est donc de $1-(p_1+p_2+p_3+p_4)$; la longueur moyenne est alors :

$$L_{\mathcal{C}} = k \times \big(1 - (p_1 + p_2 + p_3 + p_4)\big) + (k+1) \times \big(p_1 + p_2 + p_3 + p_4\big) = k + p_1 + p_2 + p_3 + p_4$$

- **(c)** oui c'est possible si les quatre lettres les plus en bas sont les quatre les moins probables et moins probables chacune que la somme de deux d'entre elles [1] et qu'ensuite toutes les autres sont aussi probables (entre elles) ; par exemple : $p_1 \leq p_2 \leq p_3 \leq p_4 < p_1 + p_2$ et $p_4 < p_5 = p_6 = ... = p_n$.
  Notez que l'équiprobabilité de toutes les lettres **ne** conduit **pas** à cet arbre pour un code de Huffman car, comme n est pair, la première étape de l'algorithme du code de Huffman les regroupera *toutes* deux par deux, et donc on n'aura pas les deux sous-arbres exterieurs à 3 feuilles, même si, cet arbre là aura en effet la même distribution de longueurs (et donc la même longueur moyenne) qu'un code de Huffman.

Commentaire : Quelques confusions sur (c).

② **[5 points]** On considère une séquence X telle que celles décrites au début. Soit N la longueur de cette séquence ($N \geq n$). Avec ce qui a été fait jusqu'ici, que pouvez-vous déjà donner comme meilleures bornes (haute et basse) pour l'entropie $H(X)$?

Justifiez votre réponse.

Réponses et justifications : Soit $L_{\mathcal{C}}$ la longueur moyenne de $\mathcal{C}$, L_{Hu} la longueur moyenne d'un code de Huffman de X et soit $L_{\text{S-F}}$ la longueur moyenne d'un code de Shannon-Fano de X.

1. sinon on a un sous-arbre à 3 feuilles ; ce qui serait possible, mais il faut alors chercher des conditions sur les cinquième et sixième feuilles ; ce qui est possible mais plus compliqué ; on ne cherche ici qu'une condition suffisante.

Par l'optimalité du code de Huffman, on sait que

$$L_{\text{Hu}} \leq L_{\mathcal{C}} \qquad \text{et} \qquad L_{\text{Hu}} \leq L_{\text{S-F}}$$

Par le théorème de Shannon, on sait que :

$$H(X) \leq L_{\text{Hu}} \qquad \text{et} \qquad L_{\text{S-F}} < H(X) + 1$$

(La première inégalité est vraie pour tout code sans-préfixe et sans perte, mais, donc, est la plus stricte pour L_{Hu} en raison de l'optimalité énoncée ci-dessus.)

On a donc :

$$0 \leq L_{\text{Hu}} - 1 \leq L_{\text{S-F}} - 1 < H(X) \leq L_{\text{Hu}} \leq L_{\mathcal{C}}$$

Comme on n'a pas parlé des codes de Shannon-Fano, ci-dessus, la réponse minimale attendue est :

$$L_{\text{Hu}} - 1 < H(X) \leq L_{\text{Hu}} \leq L_{\mathcal{C}}$$

Attention ici, le terme minorant dans l'inégalité ci-dessus utilise bien L_{Hu} et surtout **pas** $L_{\mathcal{C}}$ car il n'est pas dit que nous soyons dans le cas où $\mathcal{C}$ est un code de Huffman pour X !

Une version plus large ne parlant pas du code de Huffman serait :

$$0 \leq H(X) \leq L_{\mathcal{C}}$$

ou, sans même plus parler de code alors :

$$0 \leq H(X) \leq \log_2(n) < k + \frac{1}{2^{k-1} \ln 2} < k + 1$$

Enfin, même si ce n'est pas strictement « *ce qui a été fait jusqu'ici* », on pourrait penser aux situations extrêmes possibles pour encadrer l'entropie :

- l'entropie minimale correspond à la situation où toutes les lettres sauf une apparaissent une fois, et la dernière qui apparait $N - (n-1)$ fois ; ce qui donne comme entropie :

$$H_{\min} = \log_2(N) - \frac{N-n+1}{N} \log_2(N-n+1) = \frac{n-1}{N} \log_2(N) + \frac{N-n+1}{N} \log_2\left(\frac{N}{N-n+1}\right)$$

- l'entropie maximale correspond à la situation la plus proche de l'équiprobabilité : μ lettres qui apparaissent λ fois et $n-\mu$ lettres qui apparaissent $\lambda - 1$ fois avec $\lambda = \lceil \frac{N}{n} \rceil$ et $\mu = N \mod n$; ce qui donne comme entropie :

$$H_{\max} = \frac{\lambda\,\mu}{N} \log_2\left(\frac{N}{\lambda}\right) + \frac{(n-\mu)\,(\lambda-1)}{N} \log_2\left(\frac{N}{\lambda-1}\right)$$

Commentaire : Peu ont pas donnés les bornes optimales ici.

③ **[8.5 points]** Soit n_i le nombre d'apparitions dans X de la i-ème lettre, classées par ordre d'apparition :

$$1 \leq n_1 \leq n_2 \leq n_3 \leq n_4 \leq \ldots \leq n_{2^k} \leq n_{2^k+1} \leq n_n$$

On suppose de plus $N \geq 2\,n - 3$ et $n_4 < n_5$ (inégalité *stricte*).
Quelle est l'entropie minimale pour X (formule en fonction de n et N) ?
Justifiez *pleinement* votre réponse.

Réponse et justification : L'entropie est minimale lorsque l'on est le plus proche de la situation déterministe, c.-à-d. lorsque toutes les lettres sauf une sont à 1 et que cette dernière est au score restant maximal.

MAIS la condition $n_4 < n_5$ empêche cette situation et impose que $n_5 = 2$.

On a donc une entropie minimale pour :

$$n_1 = n_2 = n_3 = n_4 = 1 < n_5 = ... = n_{n-1} = 2 \leq n_n$$

Que vaut n_n dans ce cas ?

$$n_n = N - 4 - (n-5) \times 2 = N + 6 - 2n$$

lequel est supérieur ou égal à 3 puisque $N \geq 2\,n - 3$.

Dans ce cas, l'entropie vaut :

$$H(X_{\min}) = \log_2(N) - (n-5)\,\frac{2}{N}\log_2(2) - \frac{n_n}{N}\log_2(n_n) = \log_2(N) + \left(\frac{2n-6}{N} - 1\right)\log_2(N+6-2n) - \frac{2\,(n-5)}{N}$$

ou aussi :

$$\begin{aligned} H(X_{\min}) &= \frac{4}{N}\log_2(N) + \frac{2\,(n-5)}{N}\log_2(\frac{N}{2}) + \frac{n_n}{N}\log_2(\frac{N}{n_n}) \\ &= \frac{4}{N}\log_2(N) + \frac{2\,(n-5)}{N}\log_2(\frac{N}{2}) + \frac{N+6-2n}{N}\log_2(\frac{N}{N+6-2n}) \end{aligned}$$

Commentaire : Certain(e)s n'ont pas réussi à poser les contraintes bien que connaissant la situation d'entropie minimale d'un point de vue théorique.
Aussi plusieurs erreurs de développement.

④ **[7 points]** On suppose maintenant que $N = 3\,n - 4$ (et toujours $n_4 < n_5$).
Quelle est l'entropie maximale pour X (formule en n uniquement) ? **Justifiez** *pleinement* votre réponse.

Réponse et justification : L'entropie est maximale lorsque l'on a équirépartition. La condition $n_4 < n_5$ empêche l'équirépartition parfaite, mais la meilleure approximation est alors d'avoir :

$$n_1 = n_2 = n_3 = n_4 = n_5 - 1 < n_5 = ... = n_n$$

Combien vaut n_5 dans ce cas ? On a :

$$N = 4 \times (n_5 - 1) + (n-4) \times n_5 = n \times n_5 - 4$$

donc :

$$n_5 = \frac{N+4}{n} = \frac{3\,n - 4 + 4}{n} = 3$$

Dans ce cas, l'entropie vaut (l'une quelconque de ces formules suffit) :

$$\begin{aligned} H(X_{\max}) &= \log_2(N) - 4\,\frac{2}{N}\log_2(2) - (n-4)\frac{3}{N}\log_2(3) \\ &= \log_2(3\,n-4) - 4\,\frac{2}{3\,n-4}\log_2(2) - \frac{3\,(n-4)}{3\,n-4}\log_2(3) \\ &= \log_2(3\,n-4) - \frac{8}{3\,n-4} - \frac{3\,(n-4)}{3\,n-4}\log_2(3) \end{aligned}$$

ou aussi :

$$\begin{aligned} H(X_{\max}) &= 4\,\frac{2}{N}\log_2(\frac{N}{2}) + (n-4)\frac{3}{N}\log_2(\frac{N}{3}) \\ &= \frac{8}{3\,n-4}\log_2(\frac{3\,n-4}{2}) + \frac{3\,n-12}{3\,n-4}\log_2(\frac{3\,n-4}{3}) \\ &= \log_2(3\,n-4) - \frac{8}{3\,n-4} - \frac{3\,n-12}{3\,n-4}\log_2(3) \end{aligned}$$

Commentaire : Mêmes remarques que pour la sous-question précédente.

⑤ **[3 points]** Dans les conditions de la question ④ précédente, quelle est la longueur moyenne du code de Huffman ? (formule en k et n, ou en k seulement) **Indication :** n'oubliez pas la question ①.

Justifiez votre réponse.

Réponse et justification : Dans ce cas (④) on est exactement dans le cas décrit en ①c, et donc la longueur moyenne est :

$$L_{\text{Hu}} = k + p_1 + p_2 + p_3 + p_4 = k + \frac{8}{3n-4} = k + \frac{4}{3\cdot 2^{k-1}+1}$$

Commentaire : Peu abordé. (Pas vu le lien avec ① ?)

⑥ **[3 points]** À partir de vos réponses aux deux dernières questions, quelle inégalité algébrique pouvez-vous en tirer (une seule suffit ici) ?

Justifiez votre réponse.

Réponse et justification : On a (cf ②) :

$$L_{\text{Hu}} - 1 < H(X) \leq L_{\text{Hu}}$$

et donc :

$$k - 1 + \frac{8}{3n-4} < \log_2(3\,n-4) - \frac{8}{3\,n-4} - \frac{3\,(n-4)}{3\,n-4}\log_2(3) \leq k + \frac{8}{3n-4}$$

dont on peut donner l'une ou l'autre des inégalités.