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 MA
Doeraene S.

NOM : Hanon Ymous
(000000)
Place : 0

#0000

Information, Calcul et Communication (SMA) : Examen I

1er novembre 2024

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 de l'espace supplémentaire pour des réponses en fin de copie (page 16).
5. Vous pouvez répondre en français ou en anglais.
6. 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 ou l'une des assistantes.
7. Sauf indication contraire, on considère que les opérations arithmétiques, le calcul de la **taille** d'une liste et l'accès à un élément d'une liste sont des opérations élémentaires. De même, sauf indication contraire, on parle de complexité temporelle pire cas.
8. L'examen comporte 6 questions indépendantes sur 16 pages, qui peuvent être traitées dans n'importe quel ordre, mais qui ne rapportent pas la même chose (les points sont indiqués, le total est de **150** points).

   Toutes les questions comptent pour la note finale.

Question 1 – Questions diverses [27 points]

① **[4 points]** Soit $x = -\frac{38}{2^2}$. Comment écrit-on x en virgule flottante (binaire) avec 3 bits d'exposant et 4 bits de mantisse ? Rappel : on utilise la représentation simplifiée dans laquelle l'exposant n'a pas de biais (et est donc toujours positif ou nul) ni de comportement particulier pour 0.
Justifiez *pleinement* votre réponse.

Réponse puis justification :

Considérez l'algorithme suivant, où $\lfloor x \rfloor$ représente la partie entière de x :

chiffres

entrée : *nombre entier* p
sortie : *???*

Si $p < 0$
  Sortir : $1 +$ **chiffres**$(-p)$
Si $p < 10$
  Sortir : 1
Sortir : **chiffres**$(\lfloor \frac{p}{10} \rfloor) + 1$

② **[2 points]** Quelle est la sortie de **chiffres**(-486) ? Justifiez *brièvement* votre réponse.

③ **[3 points]** Quelle est la complexité de **chiffres** ? Justifiez *pleinement* votre réponse.

Réponses et justifications :

Considérez l'algorithme suivant :

testeListe

entrée : *liste L de nombres entiers positifs*
sortie : *oui ou non*

```
n ⟵ taille(L)
Pour i de 1 à n − 1
    Pour j de i + 1 à n
        Si L[i] > L[j]
            Sortir : « non »
Sortir : « oui »
```

④ **[2 points]** Quelle est la sortie de **testeListe** pour la liste $(2, 5, 4, 6)$?
Justifiez *brièvement* votre réponse.

⑤ **[3 points]** Quelle est la complexité de **testeListe** ? Justifiez *brièvement* votre réponse.

⑥ **[3 points]** Proposez un changement à **testeListe** pour améliorer sa complexité sans changer son résultat.

Réponses et justifications :

⑦ **[4 points]** Soient deux nombres $m, n \in \mathbb{N}$, que l'on peut stocker sur p et q bits non signés, respectivement (sans nécessairement utiliser tous les bits disponibles). De combien de bits, au maximum, a-t-on besoin pour stocker, sans dépassement de capacité, le résultat des opérations suivantes : $m \times n$, $m + n$? Justifiez *brièvement* chaque réponse (les réponses ne sont pas forcément les mêmes pour les deux opérations).

Réponses puis justifications :

---

⑧ **[2 points]** Soit n un nombre entier qui s'écrit `0100110` en binaire signé sur **7** bits. Comment s'écrit $-n$ (toujours sur 7 bits) ?
Justifiez *brièvement* votre réponse.

Réponse puis justification :

⑨ **[4 points]** On calcule sur des entiers binaires signés sur **8** bits. Que vaut le résultat (reconvertit en décimal) de $(-5) \times 46$?
Justifiez votre réponse.

Réponse puis justification :

Question 2 – Corrections d'erreurs [20 points]

Votre équipe est spécialisée dans la prévention de dangers d'origine alien. En particulier, elle prend des mesures de « zébrons » (une mesure fictive) provenant de l'espace, et cherche à détecter des anomalies.

D'après son expérience passée, les zébrons mesurés d'un jour à l'autre (des entiers strictement positifs) ne devraient pas trop varier. Au maximum, ils devraient monter ou descendre de 10 unités. Si la différence d'un jour au suivant devait être strictement supérieure à 10 unités, cela pourrait signifier que des aliens sont en approche. Dans ce cas, on veut alerter le Conseil Fédéral.

Malheureusement, les appareils ne sont pas toujours fiables. Certains jours, ils ne fonctionnent pas bien, et produisent un `0` à la place. On veut ignorer ces `0` dans les mesures, car on ne veut pas sonner l'alerte pour rien. S'il y a un ou plusieurs `0` dans les mesures, on veut comparer les mesures valides qui les entourent.

Votre collègue a écrit un programme C++ `monitoring` (au dos de cette page) qui implémente cette détection. La fonction prend en entrée un vecteur des mesures, et vérifie que les différences de jour en jour (en ignorant les `0`) ne sont pas trop importantes. Elle retourne un vecteur des jours (indices dans le vecteur d'origine) où la quantité de zébrons a trop changé par rapport à la dernière mesure valide.

Votre collègue a même écrit quelques tests pour cette fonction (au dos également), qui semble fonctionner.

Cependant, on constate en pratique que ce programme comporte plusieurs erreurs. On trouve des faux positifs (des jours rapportés alors qu'ils n'étaient pas anormaux), des faux négatifs (des jours qui ne sont pas rapportés alors qu'ils étaient anormaux) et même des crashes.

① **[9 points]** Donnez 3 nouveaux tests, qui mettent en évidence les 3 cas de problèmes constatés : un faux positif, un faux négatif, et un crash. Pour chaque test, donnez l'entrée, la sortie *attendue* et la sortie *du programme* (cette dernière étant donc un crash dans le dernier cas).

② **[11 points]** Indiquez et **corrigez** toutes les erreurs (il peut y en avoir plus que 3). **Marquez** les erreurs directement dans le code, puis **corrigez-les** sur la page d'en face, en regard. Mettez bien en évidence quelle correction correspond à quelle(s) ligne(s) incorrectes. **On ôtera 1 point pour toute indication d'une erreur qui n'en est pas une.**

En bonus : suggérez une amélioration du code (qui n'est pas, strictement parlant, une erreur) qui permettrait d'éviter à l'avenir qu'une des erreurs que vous avez corrigées ne soit réintroduite.

Pas de réponse ici ! Les réponses sont à écrire deux pages plus loin, en regard du code de votre collègue.

suite au dos ☞

Tests de votre collègue :

```
1   void testMonitoring() {
2       assert(monitoring(vector<int> { 11, 15, 22, 13, 20, 12 })
3              == (vector<size_t> {}));
4       assert(monitoring(vector<int> { 11, 15, 22, 11, 20, 12 })
5              == (vector<size_t> { 3 }));
6       assert(monitoring(vector<int> { 11, 15, 0, 11, 20, 12 })
7              == (vector<size_t> { }));
8       assert(monitoring(vector<int> { 11, 15, 22, 0, 11, 20, 8 })
9              == (vector<size_t> { 4, 6 }));
10  }
```

Fonction de votre collègue :

```
1   // omis: les #include et le using namespace std;
2
3   // Quelle différence de mesure on accepte entre deux jours
4   constexpr int TOLERANCE = 10;
5
6   vector<size_t> monitoring(vector<int> mesures) {
7       vector<size_t> resultat;
8
9       for (size_t i = 0; i + 1 < mesures.size(); i++) {
10          int elemCourant = mesures.at(i);
11          int elemSuivant = mesures.at(i + 1);
12          // note/rappel : mesures.at(i) peut être vu comme mesures[i]
13
14          if (elemCourant == 0) {
15              // On a déjà traité ce cas au tour précédent -> ignorer
16          } else if (elemSuivant != 0) {
17              // Il faut comparer les deux éléments
18              if (abs(elemSuivant - elemCourant) >= TOLERANCE) {
19                  resultat.push_back(i + 1);
20              }
21          } else {
22              // L'élément suivant est un 0
23              // donc il faut tester avec le suivant-suivant
24              int elemSuivantSuivant = mesures.at(i + 2);
25              if (abs(elemSuivantSuivant - elemCourant) > TOLERANCE) {
26                  resultat.push_back(i + 2);
27              }
28          }
29      }
30
31      return resultat;
32  }
```

Vos nouveaux tests :

Vos corrections du code :

Ne pas écrire dans cette zone.

Question 3 – Algorithmes [22 points]

Note : lisez la sous-question ④ avant de répondre à ①.

On s'intéresse au problème de la plus longue sous-séquence commune (PLSC). Étant données deux listes X et Y, on veut trouver la longueur l d'une plus longue liste L qui soit une « sous-séquence » de X et de Y. Par sous-séquence de X, on entend que L ne doit contenir que des éléments de X, dans le même ordre, mais pas forcément *tous* les éléments de X. Autrement dit, on peut passer de X à L uniquement en supprimant des éléments. L'algorithme recherché ne doit pas trouver L elle-même, mais seulement sa *longueur* l.

Par exemple, pour les listes $X = (1, 2, 3, 2, 4, 1, 2)$ et $Y = (2, 4, 3, 1, 2, 1)$, la liste $(2, 3, 1)$ est une sous-séquence commune de X et Y. En revanche, ce n'est pas *la plus longue*. Les sous-séquences $(2, 3, 2, 1)$ et $(2, 4, 1, 2)$ sont toutes deux des sous-séquences de X et de Y mais sont de longueur 4 et sont les plus longues telles sous-séquences. Le résultat l doit donc être 4.

On peut trouver l avec la fonction f suivante (la preuve sort du cadre de ce cours/examen) :

$$f(i,j) = \begin{cases} 0 & \text{si } i = 0 \text{ ou } j = 0 \\ f(i-1, j-1) + 1 & \text{si } i, j > 0 \text{ et } X[i] = Y[j] \\ \max(f(i, j-1), f(i-1, j)) & \text{si } i, j > 0 \text{ et } X[i] \neq Y[j] \end{cases}$$

l est alors $f(\textbf{taille}(X), \textbf{taille}(Y))$.

① **[8 points]** Écrivez un algorithme **PLSCAnnexe**(X, Y, i, j) *récursif, sans boucle*, qui calcule $f(i, j)$ pour des listes X et Y données. N'oubliez pas de bien spécifier les préconditions (rappel : les contraintes sur les valeurs possibles des paramètres) nécessaires à ce que le problème soit bien défini.

Réponse :

② **[1 points]** Écrivez un algorithme **PLSC**(X, Y) qui répond au problème de la plus longue sous-séquence commune de X et Y. Vous pouvez bien sûr réutiliser l'algorithme **PLSCAnnexe** que vous venez de définir.

Réponse :

suite ☞

③ [**5 points**] Quelle est la complexité de votre algorithme proposé en ② ? **Justifiez** votre réponse.

Vous pouvez supposer que X et Y ont la même taille.

Conseil : commencez par bien identifier quel est le *pire cas* pour votre algorithme.

Réponse et justification :

④ [**8 points**] Écrivez un algorithme (avec ou sans boucles) en $\Theta(n^2)$ maximum, où n est la taille (supposée commune) de X et Y. Justifiez pourquoi il est en $\Theta(n^2)$.

Si votre réponse à ② est déjà en $\Theta(n^2)$, alors vous n'avez rien de plus à faire ici que le dire (et recevrez une note sur la somme des points des deux sous-questions).

Réponse :

Question 4 – Machine de Turing [16 points]

On considère la machine de Turing ayant pour table de transition :

	0	1	ε
1	$(2, \varepsilon, +)$	$(1, \varepsilon, +)$	$(4, 1, -)$
2	$(3, \varepsilon, +)$	$(2, \varepsilon, +)$	$(5, 1, -)$
3	$(1, \varepsilon, +)$	$(3, \varepsilon, +)$	$(5, 1, -)$
4	$(7, 0, -)$	$(7, 1, -)$	$(6, 1, -)$
5	$(7, 0, -)$	$(7, 1, -)$	$(6, 0, -)$
6	$(7, 0, -)$	$(7, 1, -)$	$(7, \varepsilon, +)$

① **[5 points]** Quel est l'état de la bande et la position de la tête de lecture lorsque la machine s'arrête, si elle a démarré dans l'état 1 avec sa tête de lecture positionnée comme suit :

Conseil : travaillez d'abord au crayon et à la gomme pour « exécuter » la machine.

$\cdots\varepsilon$	0	1	0	0	1	0	$\varepsilon\cdots$
	↑						

② **[11 points]** Dites en une phrase ce que fait cette machine, puis justifiez votre réponse en quelques phrases.

Réponses :

Question 5 – Sommes nulles [35 points]

Le problème SSP est un problème de décision sur un ensemble de nombres entiers, qui cherche à déterminer s'il existe un sous-ensemble non-vide de somme nulle.

Par exemple, pour l'ensemble $\{3, 4, 12, -1, 6, -2\}$ la réponse est « oui » puisque le sous-ensemble $\{3, -1, -2\}$ somme bien à 0.
Par contre, pour l'ensemble $\{3, 2, 14, -8, 1\}$ la réponse est « non » puisqu'aucun de ses sous-ensembles de valeurs ne somme à 0.

SSP est connu pour être dans NP. Mais qu'est-ce que cela veut dire concrètement ? Nous allons regarder cela d'un peu plus près.

Note : si cela vous aide, vous pouvez, sans perte de généralité, considérer qu'un ensemble est une liste.

① **[2 points]** Pour commencer, pour écrire des algorithmes sur des sous-ensembles d'un ensemble, il est parfois plus simple de voir un sous-ensemble comme l'écriture binaire d'un entier positif : `1` si l'élément est présent dans le sous-ensemble, et `0` s'il ne l'est pas

Par exemple, dans l'exemple ci-dessus ($\{3, 4, 12, -1, 6, -2\}$), le sous-ensemble $\{3, -1, -2\}$ serait représenté par le nombre binaire `100101` : 3 est présent, 4 et 12 ne le sont pas, -1 est présent, 6 ne l'est pas et -2 l'est.

À quelle valeur en base 10 correspond `100101` ? Justifiez *brièvement* votre réponse.

Réponse et justification :

② **[10 points]** En supposant qu'il existe un algorithme **somme** qui prend en entrée un ensemble E et un nombre entier k positif non nul et qui donne en sortie la somme des éléments de E correspondant au sous-ensemble de E décrit par l'écriture binaire de k, proposez un algorithme permettant de répondre au problème SSP.

Réponse :

suite au dos ☞

③ [**3 points**] En supposant que la complexité de **somme** est en $\Theta(n)$ où n est la taille de E, quelle est la complexité de l'algorithme que vous avez proposé en ② ? **Justifiez** votre réponse.

④ [**3 points**] Cette complexité est-elle en contradiction avec l'affirmation que SSP est dans NP ? Détaillez votre réponse.

Réponses et justifications :

⑤ [**3 points**] Quel serait un certificat pour une instance (positive) de ce problème ?

Illustrez votre explication dans le cas concret donné en exemple au début.

Réponse :

suite ☞

⑥ **[8 points]** Proposez un algorithme qui, recevant un ensemble et un certificat tel que vous l'avez défini en ⑤, permet de vérifier qu'une instance positive de SSP en est bien une.

Vous **ne** pouvez **pas** utiliser l'algorithme **somme** ici.

Réponse :

⑦ **[3 points]** Quelle est la complexité de l'algorithme que vous avez proposé en ⑥ ? Justifiez votre réponse.

⑧ **[3 points]** Cette complexité est-elle en contradiction avec l'affirmation que SSP est dans NP ? Détaillez votre réponse.

Réponses et justifications :

Question 6 – Développement en série [30 points]

On souhaite implémenter nous-mêmes, en C++, le calcul du logarithme naturel. Pour ce faire, nous allons nous baser sur le développement en série de Maclaurin de $\ln(1+x)$, qui est :

$$\ln(1+x) = \sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{i} x^i$$

Puisque nous ne pouvons pas réellement aller jusqu'à ∞, nous allons limiter la série jusqu'à un entier n donné, et donc calculer une *approximation* de la fonction jusqu'au rang n :

$$\ln(1+x) \approx f_n(x) = \sum_{i=1}^{n} \frac{(-1)^{i+1}}{i} x^i$$

Comment savoir quelle valeur de n est suffisamment grande pour obtenir une approximation suffisamment bonne ? Nous allons écrire un programme pour le déterminer.

Important ! Dans cette question, vous n'avez pas le droit d'utiliser les fonctions mathématiques de C++, sauf là où c'est explicitement autorisé ! Vous ne pouvez utiliser que les opérations élémentaires sur les entiers et les nombres à virgule flottante. C'est logique, puisqu'on cherche à implémenter ln nous-mêmes.

Dans chaque sous-question, vous pouvez utiliser les fonctions définies pour les sous-questions précédentes.

① **[5 points]** Écrire (en C++) la fonction **intPow** qui prend en entrée un réel x et un entier positif ou nul y et qui retourne x^y.

Réponse :

② **[10 points]**

Écrire (en C++) la fonction **log1pApprox** qui prend en entrée un réel x et un entier strictement positif n, et qui retourne la valeur de $f_n(x)$ (l'approximation au rang n de $\ln(1+x)$ avec le développement en série de Maclaurin). Vous devez respecter la définition de $f_n(x)$ ci-dessus.

suite ☞

Réponse :

③ **[10 points]** Il faut maintenant pouvoir tester si un n donné est « suffisamment bon ». Écrire (en C++) la fonction **suffisammentBon** qui prend en entrée un entier strictement positif n et un réel strictement positif `tolerance`, et qui retourne `true` si et seulement si n est « suffisamment bon ».

Nous dirons que n est suffisamment bon si $|f_n(x) - \ln(1+x)| \leq$ `tolerance` pour tous les $x \in [0, 1[$ (0 inclus, 1 exclu). En pratique, on testera seulement les valeurs par incrément de $1/256$ (donc, $x \in \{0, 1/256, 2/256, \ldots, 255/256\}$).

Pour calculer la valeur « exacte » de $\ln(1+x)$, utilisez la fonction C++ `std::log1p`(x) (ceci est donc la seule fonction mathématique de C++ que vous avez le droit d'utiliser).

Réponse :

suite au dos ☞

④ [**5 points**] Finalement, nous voulons trouver le plus petit n qui nous donne une fonction f_n avec une tolérance donnée. Écrire (en C++) une fonction **main** qui affiche le plus petit entier strictement positif n tel que n soit « suffisamment bon » avec la tolérance 1/1000).

Réponse :

Place supplémentaire pour répondre à n'importe quelle question si nécessaire.
Mais VEUILLEZ INDIQUER LE NUMÉRO DE LA QUESTION TRAITÉE.