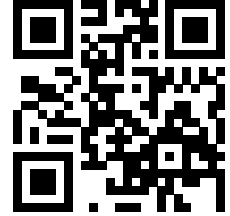


NOM : Hanon Ymous
(000000)
Place : 0

#0000



Information, Calcul et Communication (SMA/SPH) :

Examen I

3 novembre 2023

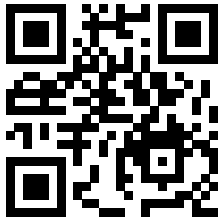
SUJET 1

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 7 exercices indépendants sur 16 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 145 points).

Tous les exercices comptent pour la note finale.



Question 1 – Calcul approché. [23 points]

On cherche à écrire un programme C++ permettant de calculer une approximation du nombre $\frac{1}{e}$, en utilisant pour cela le développement en série entière de l'exponentielle :

$$\frac{1}{e} = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!}$$

Note : $0! = 1$ et $(-1)^0 = 1$.

① [5 points] Écrire en C++ une fonction `factorielle()` qui prend en entrée un entier n sous forme de `double` et renvoie $n!$ (sous forme de `double`). Il n'est pas demandé de vérifier que la valeur n reçue en argument est bien une valeur entière.

Réponse :

② [9 points] On s'intéresse maintenant à l'approximation à l'ordre n de $\frac{1}{e}$: $\sum_{k=0}^n \frac{(-1)^k}{k!}$. Écrire une fonction `approx_inv_e()` qui prend en entrée un entier n et renvoie l'approximation de $\frac{1}{e}$ à l'ordre n en utilisant la formule ci-dessus.

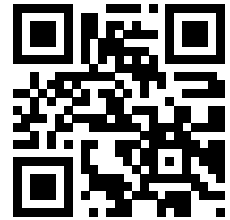
Vous veillerez à minimiser la complexité de l'algorithme utilisé. (Voir aussi la question suivante.)

Réponse :

Ne pas écrire dans cette zone.

Question 1


Anonymisation : #0000
p. 3

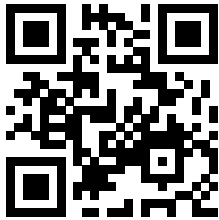


(suite de la réponse :)

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

Réponse et justification :

suite au dos 



④ [6 points] Écrire une fonction `affiche_inv_e()` qui demande à l'utilisateur d'entrer un entier n strictement supérieur à 1 (répéter la question tant que ce n'est pas le cas), et qui affiche successivement l'approximation de $\frac{1}{e}$ à l'ordre 2, 3, ..., n .

Exemple de déroulement :

Entrez un entier strictement supérieur à 1 : 0

Entrez un entier strictement supérieur à 1 : 5

ordre 2 : 0.5

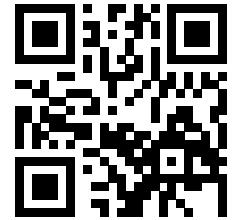
ordre 3 : 0.333333

ordre 4 : 0.375

ordre 5 : 0.366667

Réponse :

Ne pas écrire dans cette zone.



Question 2 – Quelques algorithmes. [29 points]

Note : lisez la sous-question ③ (au dos) avant de répondre à ①.

Soient L_1 et L_2 deux listes quelconques de nombres entiers tous différents les uns des autres (dans chaque liste ; pas entre les listes). On cherche à écrire un algorithme ayant pour entrée ces deux listes et dont la sortie soit leur intersection (la liste des valeurs en commun), dans un ordre quelconque.

Par exemple, pour $L_1 = (-3, 12, 5, -2, 1, -16)$ et $L_2 = (7, 1, -9, 5, 4)$, un tel algorithme sortira alors p.ex. $(5, 1)$ (l'ordre peut être différent).

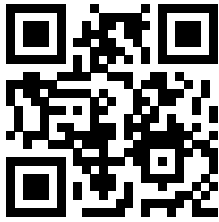
① **[6 points]** Écrivez un algorithme pour résoudre ce problème.

Réponse :

② **[2 points]** Dans le cas où les deux listes ont la même taille n , quelle est la complexité de votre algorithme proposé en ① ? **Justifiez** votre réponse.

Réponse et justification :

suite au dos 



③ [6 points] Écrivez un algorithme de complexité temporelle *strictement* moindre que $\Theta(n^2)$ pour résoudre le problème proposé.

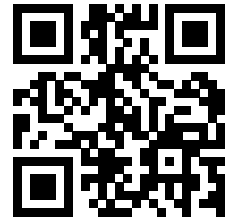
Si votre réponse à ① est déjà en une complexité *strictement* moindre que $\Theta(n^2)$ lorsque les deux listes ont la même taille n , alors vous n'avez rien de plus à faire ici (et serez, bien entendu, noté(e) sur la somme des points des deux sous-questions).

Réponse :

④ [3 points] Dans le cas où les deux listes ont la même taille n , quelle est la complexité de votre algorithme proposé en ③? **Justifiez** votre réponse.

Réponse et justification :

Ne pas écrire dans cette zone.



⑤ [8 points] On suppose maintenant que les deux listes sont *triées*.
Écrivez un algorithme *récuratif* et sans boucle pour résoudre le problème proposé.

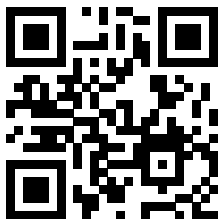
Par exemple, pour $L_1 = (-16, -3, -2, 1, 5, 12)$ et $L_2 = (-9, 1, 4, 5, 7)$, un tel algorithme sortira alors p.ex. (1, 5) (l'ordre peut être différent).

Réponse :

⑥ [4 points] Dans le cas où les deux listes ont la même taille n , quelle est la complexité de votre algorithme proposé en ⑤? **Justifiez** votre réponse.

Réponse et justification :

suite au dos 



Question 3 – Tournez machines. [16 points]

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

	0	1	ε
1	(3, ε , +)	(2, ε , +)	(8, 1, +)
2	(2, ε , +)	(2, ε , +)	(8, 0, +)
3	(3, 0, +)	(3, 1, +)	(4, ε , -)
4	(6, ε , -)	(5, ε , -)	(8, 0, +)
5	(5, 0, -)	(5, 1, -)	(1, ε , +)
6	(6, ε , -)	(6, ε , -)	(8, 0, +)

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 avec sa tête de lecture positionnée comme suit :

$\dots\varepsilon$	0	0	0	0	1	1	1	1	$\varepsilon\dots$
	↑								

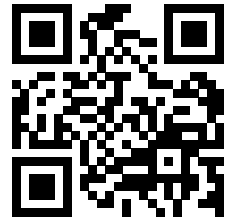
② [11 points] Justifiez votre réponse en trois ou quatre phrase(s), puis dites en une phrase ce que fait cette machine.

Réponses :

Ne pas écrire dans cette zone.


Question 3

Anonymisation : #0000
p. 9



(suite de la réponse :)

Ne pas écrire dans cette zone.

suite au dos 



Question 4 – Satisfaits ? [29 points]

Le problème SAT (pour satisfaisabilité booléenne) est un problème de décision sur une formule de logique booléenne, qui cherche à déterminer s'il existe une assignation des variables telle que la formule soit vraie.

Par exemple (donné ici en C++ pour faciliter la lisibilité et la compréhension) :

```
bool x1(false);    bool x2(true);    bool x3(false);  
  
bool f1( (x1 and x2) or (not x3) );
```

La formule `f1` est *satisfaite* (c.-à-d. est vraie) pour l'assignation des variables `x1`, `x2`, `x3` faite ci-dessus. Donc la réponse à cette instance de SAT est « oui ».

En revanche, la formule

```
bool f2( x1 and x2 and (not x1) );
```

n'est pas satisfiable : aucune affectation de la variable `x1` ne peut la rendre vraie (on dit « la *satisfaire* »). Donc la réponse à cette autre instance de SAT est « non ».

On s'intéresse ici à la résolution algorithmique de ce problème SAT.

① [2 points] Quelles sont la ou les entrée(s) d'un algorithme résolvant ce problème ?
Quelles sont sa ou ses sortie(s) ?

Entrée(s) :

Sortie(s) :

② [3 points] Qu'est-ce qu'un certificat pour une instance (positive) de ce problème ?

Réponse :

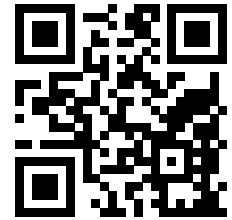
③ [2 points] Si on a une formule constituée uniquement de `or` entre de simples variables (comme p.ex. `x1 or x2 or x3 or x4`) et une assignation de ses variables, quelle est la complexité du calcul de sa valeur (vraie ou fausse) ? **Justifiez** votre réponse.

Réponse et justification :

④ [2 points] Si on a une formule constituée uniquement de `and` entre de simples variables (comme p.ex. `x1 and x2 and x3`) et une assignation de ses variables, quelle est la complexité du calcul de sa valeur (vraie ou fausse) ? **Justifiez** votre réponse.

Réponse et justification :

Ne pas écrire dans cette zone.



⑤ [6.5 points] D'après les deux questions précédentes, et sachant que toute formule booléenne peut être transformée en une séquence, de taille linéaire, de **and** ne portant chacun que sur des séquences (de taille linéaire) de **or** sur de simples variables¹, que pouvez-vous dire de la classe de complexité de ce problème ? **Justifiez pleinement** votre réponse.

Réponse et justification :

⑥ [2.5 points] Pour écrire des algorithmes sur des assignations de variables booléennes, il est plus simple de voir de telles assignations comme l'écriture binaire d'un entier positif.

Si on a par exemple cinq variables booléennes x_1, \dots, x_5 et que l'on fait correspondre x_1 au bit de poids fort (et x_5 au bit de poids faible), quelle serait leur assignation correspondant à l'entier 13 ?

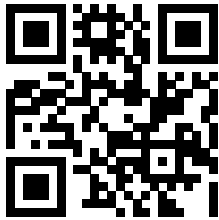
Justifiez votre réponse.

Note : une assignation est simplement une liste de valeurs booléennes. Par exemple l'assignation donnée dans le premier exemple tout au début est simplement la liste (`false, true, false`).

Réponse et justification :

suite au dos 

1. On parle de transformation linéaire en forme normale conjonctive équisatisfiable.



⑦ [6 points] En supposant que vous ayez :

- un algorithme « **assignation** », qui prend en entrée un nombre entier positif et une taille n , et qui sorte une assignation de n variables booléennes correspondant aux bits de l'écriture binaire de ce nombre (cf question précédente),
 - un algorithme « **évalue** » qui prend en entrée une formule logique et une assignation de ses variables et qui, en sortie, donne la valeur (vraie ou faux) de la formule pour cette assignation,
- proposez un algorithme qui, étant donné une formule logique, renvoie une assignation des variables satisfaisant la formule s'il en existe une, ou la liste vide sinon.

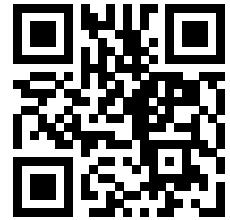
Note : vous pouvez considérer que l'on connaît le nombre de variables d'une formule logique (exactement comme on connaît la taille d'une liste) : $n \leftarrow \mathbf{nb_var}(\text{formule})$.

Réponse :

⑧ [5 points] Quelle est la complexité de votre algorithme proposé en ⑦ ?
Justifiez *pleinement* votre réponse.

Réponse et justification :

Ne pas écrire dans cette zone.



Question 5 – Un drôle de calcul. [8 points]

Considérez le code C++ suivant :

```
int g(int n, int x, int y)
{
    if (n <= 0) return x;
    return g(n-1, x+y, y);
}

int f(int n)
{
    return g(n, 0, 3);
}
```

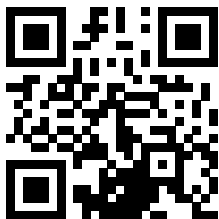
- ① [1 point] Quelle est la valeur de $f(3)$? **Réponse :**
- ② [1 point] Quelle est la valeur de $f(5)$? **Réponse :**
- ③ [1 point] Quelle expression mathématique est réalisée par cette fonction ?
[2 points] Justifiez votre réponse par une brève démonstration.

Réponse et démonstration :

- ④ [1 point] Quelle est la complexité de l'algorithme implémenté par $f()$?
[2 points] Justifiez votre réponse.

Réponse et justification :

suite au dos



Question 6 – Mauvais calcul ! [21 points]

Voici un programme C++ visant à implémenter deux algorithmes permettant de calculer a^n , pour $a \in \mathbb{R}$ et $n \in \mathbb{N}$:

```
1  #include <iostream>
2  using namespace std;
3
4  double exp_rec(double a, int n)
5  {
6      double res;
7      if (n <= 0) res = 0;
8      if (n == 1) res = a;
9      if (n%2 == 0) {
10         res = exp(a*a, n/2);
11     } else {
12         res = exp(a*a, n/2) * a;
13     }
14     return res;
15 }
16
17 double exp_iter(double a, int n)
18 {
19     double res;
20     while (n >= 0) {
21         if (n%2 == 1) {
22             res *= a;
23         }
24         a = a*a;
25         n = n/2;
26     }
27     return res;
28 }
29
30 int main()
31 {
32     double a;
33     int n;
34     cout << "Entrez un réel a : ";
35     cin >> a;
36     cout << "Entrez un entier n : ";
37     cin >> n;
38     cout << "(rec) a^n = " << exp_rec(a,n) << endl;
39     cout << "(iter) a^n = " << exp_iter(a,n) << endl;
40     return 0;
41 }
```

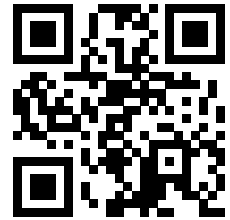
Mais ce programme comporte plusieurs erreurs de programmation, possiblement de différente nature (syntaxe, déroulement, conception, méthodologie, ...).

Indiquez et **corrigez** toutes les erreurs (directement sur le code ci-dessus).

Expliquez *brèvement* les erreurs/corrections à droite du code ou sur la page ci-contre.


On ôtera 1 point pour toute indication d'une erreur qui n'en est pas une.

Ne pas écrire dans cette zone.



Explications :

Ne pas écrire dans cette zone.

suite au dos 



Question 7 – Un peu de calcul. [19 points]

① [5 points] Quelle est la valeur décimale du nombre binaire représenté en virgule flottante sur 10 bits avec (dans cet ordre) 1 bit de signe, 3 bits d'exposant et 6 bits de mantisse, par 0100100110 ?

Justifiez brièvement votre réponse (p.ex. en explicitant vos calculs).

Réponse et justification :

② [5 points] Le nombre 2.875 (en base 10) peut-il être représenté exactement (sans erreur d'arrondi) en binaire à virgule flottante avec 2 bits d'exposant et 4 bits de mantisse ? **Justifiez** votre réponse.

Réponse et justification :

③ [4.5 points] Combien vaut, en décimal, le résultat de l'opération en binaire signé par complément à deux, $11001001 + 10101101$? **Justifiez** votre réponse.

Réponse et justification :

④ [4.5 points] Si les `int` sont stockés sur 8 bits en complément à deux, quel est le schéma binaire en mémoire de la valeur de la variable `i` suivante : `int i(-54);`

Justifiez brièvement votre réponse (p.ex. en explicitant vos calculs).

Réponse et justification :

Ne pas écrire dans cette zone.