ICC MT & EL **Test N° 1** / White **vendredi 2 novembre 2018**

Ne PAS retourner ces feuilles avant d'en être autorisé!

Merci de poser votre carte CAMIPRO en évidence sur la table.

Vous pouvez déjà compléter et lire les informations ci-dessous:

NOM en MAJUSCULES_____

Prénom _____

Numéro SCIPER _____

Signature _____

BROUILLON : Ecrivez aussi votre NOM-Prénom sur la feuille de brouillon fournie. Toutes vos réponses doivent être sur cette copie d'examen. Les feuilles de brouillon sont ramassées puis détruites.

Le test écrit commence à: **14h15**

Retourner les feuilles avec la dernière page face à vous à : **15h45**

les contrôles écrits ICC sont SANS document autorisé, ni appareil électronique

Total sur 20 points = 12 points pour la partie Quizz et ~~8~~ 13 points pour les questions ouvertes

La partie Quizz (QCM) comporte 12 questions : chaque question n'a qu'une seule réponse correcte parmi les 4 réponses proposées. Chaque réponse correcte donne 1 point. Il n'y a pas de pénalité en cas de mauvaise réponse. Aucun point n'est donné en cas de réponses multiples, de rature, ou de réponse incorrecte. Vous pouvez utiliser un crayon à papier et une gomme.

Indiquez vos réponses à la partie Quizz dans ***le tableau en bas de cette page***.

La partie « question ouverte » comporte 2 questions. Chaque question rapporte 4 points.

	Questions du Quizz												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
A													**A**
B													**B**
C													**C**
D													**D**

QUIZZ

Dans cet examen, L'opérateur **modulo** calcule le reste dans la division entière de l'opérande de gauche par l'opérande de droite et le caractère **/** désigne l'opérateur de la division entière.
Toutes les questions portant sur l'ordre de complexité travaillent sur des nombres correspondant au cadre classique pour lequel les opérateurs arithmétiques ont un coût constant quelle que soit l'amplitude des opérandes.

Question 1: Soit X et Y deux entiers non-signés sur 16 bits. Dans cette question nous utilisons l'hexadécimal pour représenter ces nombres sous une forme compacte. Sachant que X est représenté par $CAFE_{16}$ quelle valeur doit avoir Y en hexadécimal pour que le résultat de X + Y donne $FADA_{16}$?

A $2FDB_{16}$
B $3FDC_{16}$
C $2FDC_{16}$
D $2EDC_{16}$

Question 2: Soit le nombre positif Z, de motif binaire 00010111, représenté en virgule fixe non-signée sur 8 bits, avec 5 bits de partie entière et 3 bits de partie fractionnaire.
Quelle est la valeur décimale du résultat de l'expression : **8*Z - 1**

A $10,5_{10}$
B 22_{10}
C 23_{10}
D $21,98_{10}$

Question 3: On dispose d'une capacité de 10 bits pour représenter des nombres entiers signés avec la représentation du complément à deux. Quelle est la proposition correcte ?

A Le domaine couvert par cette représentation en décimal est [-512, 512]
B Les deux motifs binaires 1000000000 et 0000000000 représentent zéro
C Le plus petit nombre décimal représentable est -1000000000_{10}
D L'opposée du motif binaire 1000000000 est égal à lui-même.

Question 4: Quel est le meilleur ordre de complexité pour un algorithme de recherche de N valeurs quelconques dans une liste triée de taille N.

A $O(N^2)$ mais pas O(N log(N))
B O(N log(N)) mais pas O(N)
C O(N) mais pas O(log(N))
D O(log(N)) mais pas O(1)

Question 5: Supposons que pour des données de taille N, la résolution d'un problème PA a un ordre de complexité en $O(2^N)$ tandis que celle du problème PB a un ordre de complexité en $O(N^3)$. Quelle réponse est correcte ?

A PA est dans NP et PB est dans P
B PB est non-déterministe polynomial et PA est dans P
C PA est non-polynomial s'il n'est pas possible de vérifier une solution avec un algorithme de complexité polynomiale en fonction de N.
D Aucune des autres réponses

Question 6: Une représentation en virgule flottante exploite 2 bits pour l'exposant (toujours positif) et 4 bits pour la mantisse. La forme dénormalisée est utilisée pour la représentation des nombres ayant la valeur la plus faible de l'exposant tandis que la forme normalisée est utilisée pour les autres valeurs de l'exposant. Le domaine couvert par cette représentation en décimal est:

A [0.0625 , 15]
B [0 , 15.75]
C [0, 15.5]
D [1, 15.5]

Question 7: Soit une liste triée non-vide L contenant N entier relatifs. Voici plusieurs exemples d'une telle liste : L1={2, 4, 4, 5, 33, 33, 33, 60}, L2={-100, -77, -12, 0}, L3= {-12, -4, 1, 3, 3, 7, 9, 9, 16, 20}.

Un algorithme de recherche AlgABS(L,N,X) reçoit en entrée une telle liste L d'entiers relatifs, la taille N de cette liste et un nombre entier positif X. AlgABS(L,N,X) renvoie VRAI si X est égal à la valeur absolue d'un nombre présent dans L, et renvoie FAUX sinon.

Exemples : AlgABS(L1,8,5) renvoie VRAI, AlgABS(L2,4,9) renvoie FAUX, et AlgABS(L3,10,4) renvoie VRAI. Quel est l'ordre de complexité du meilleur algorithme qui réalise cette tâche ?

A O(N + log(N))
B O(N log(N))
C O(N)
D O(log(N))

Question 8: Les deux algorithmes **AlgoA** et **AlgoB** doivent calculer la $N^{ième}$ puissance entière d'un nombre a. Les deux données a et N sont fournies en entrée et la sortie doit être a^N. Que peut-on dire de ces deux algorithmes et de leur ordre de complexité en fonction de **N** ?

AlgoA

entrée: nombre **a**, entier **N** ≥ 0
sortie : a^N

```
v ← a
Pour i de 1 à N
    v ←  v * a
Sortir: v
```

AlgoB

entrée: nombre **a**, entier **N** ≥ 0
sortie : a^N

```
Si N = 0
    Sortir : 1
Sinon :
    v ←  AlgoB(a, N-1) ;
    Sortir v
```

A Les deux algorithmes sont corrects et ils ont un ordre de complexité différent
B **AlgoA** est correct, **AlgoB** est incorrect et ils ont le même ordre de complexité
C **AlgoA** est incorrect, **AlgoB** est correct et ils ont le même ordre de complexité
D Les deux algorithmes sont incorrects

Question 9: La *programmation dynamique* désigne une approche de conception d'algorithme dans laquelle l'algorithme…

A ne remet jamais en question un choix effectué à une étape du calcul
B dispose d'une information globale lui permettant d'effectuer le meilleur choix
C mémorise des calculs intermédiaires afin de ne pas les recalculer ultérieurement
D prend toujours une décision sur la base d'une connaissance locale facile à calculer

Question 10: Algorithme récursif (2 questions)

Compléter l'algorithme récursif suivant pour le calcul de la $N^{ième}$ puissance entière d'un nombre a. Les deux données a et N sont fournies en entrée et la sortie doit être a^N. Quelles instructions faut-il écrire aux emplacements ① et ② dans ce pseudocode ?

AlgoRec
entrée : nombre **a**, entier **N** ≥ 0 sortie : a^N
Si N = 0 ***Sortir*** 1 ***Si*** N = 1 ***Sortir*** a ***Si*** (N **modulo** 2)=1 ①?? ***Sinon*** ②??

A ① ***Sortir*** **AlgoRec**(a * a, (N – 1) / 2) * a ② ***Sortir*** **AlgoRec(**a * a, N / 2)

B ① ***Sortir*** **AlgoRec**(a * a, (N + 1) / 2) ② ***Sortir*** **AlgoRec(**a * a, (N+2) / 2)

C ① ***Sortir*** **AlgoRec**(a , (N – 3) / 2) * a ② ***Sortir*** **AlgoRec(**a , (N -2) / 2) * a

D ① ***Sortir*** **AlgoRec**(a , (N – 1) / 2) * a ② ***Sortir*** **AlgoRec(**a , N / 2) * a

Question 11: Pour votre choix à la question précédente, quel est l'ordre de complexité de **AlgoRec** ?

A O(N log N) mais pas O(N)

B O(N) mais pas O(log(N))

C O(log N) mais pas O(1)

D $O(2^N)$ mais pas $O(N^2)$

Question 12 : En 1995 Sloane et Plouffe ont défini la superfactorielle Sf(N) pour l'entier N>0 comme suit : Sf(N) = 1 ! * 2 ! *… * (N-1) ! * N !

Quel est l'ordre de complexité du meilleur algorithme pour effectuer ce calcul ?

A $O(N^N)$ mais pas $O(2^N)$

B $O(2^N)$ mais pas $O(N^2)$

C $O(N^2)$ mais pas O(N)

D O(N) mais pas O(log(N))

Questions Ouvertes

Question 1 (7 points): *Ecrire deux algorithmes exploitant* ***une liste triée non-vide L d'entiers relatifs***

Pour toutes les questions d'écriture de pseudocode, vous pouvez si vous en avez besoin définir et manipuler une (ou plusieurs) liste(s) supplémentaire(s) de la taille que vous voulez (indiquer cette taille). Cependant, excepté pour 1.2.4, il n'est pas autorisé d'« appeler » un algorithme pour manipuler une liste en une seule instruction. Par exemple, une instruction telle que **L' ← tri_fusion(L)** n'est pas autorisée.

1.1) Recherche en valeur absolue (3points)

1.1) Ecrire le pseudocode de l'algorithme qui reçoit en entrée la liste L , sa taille N>0, et un nombre entier positif X. La sortie de l'algorithme est le booléen indiquant si X est égal à **la valeur absolue** d'un nombre présent dans L. Exemple : l'algorithme renvoie **VRAI** s'il est appelé avec la liste L contenant les éléments {-5, -3, -1, 6, 9} et la valeur 3 pour X.

Contrairement à la question du quizz, on n'exige pas le « meilleur » algorithme qui résout ce problème mais seulement un algorithme qui détaille chaque instruction et obtient le résultat recherché.

1.1.1) indiquer en une à deux phrases le principe/l'ébauche de votre algorithme (0.5pt)

1.1.2) donner le pseudocode ci-dessous (2 pt)

Recherche_valeur_absolue
entrée : liste triée non-vide L d'entiers relatifs, taille de la liste N, entier positif X sortie : booléen indiquant si X est égal à la valeur absolue d'un élément de L

1.1.3) Justifier (avec une à deux phrases) l'ordre de complexité de votre algorithme en fonction de N avec la notation de Landau en O(…). (0. 5 pt)

1.2) **Recherche de l'élément le plus fréquent (4 points)**

On continue ici à travailler avec une liste triée non-vide **L** contenant N entier relatifs. De plus, on suppose que toutes les valeurs de **L** sont comprises dans l'intervalle [-MAX, MAX]. Dans cet exercice on veut trouver l'élément qui apparaît le plus souvent dans la liste. Si deux valeurs ont le même nombre d'apparitions, l'algorithme peut renvoyer n'importe laquelle des valeurs ayant le nombre maximum d'apparition.

Exemple : si l'algorithme est appelé avec la liste L = {-5, -5,-3, -3, -3, -3, -1, 2, 2, 2, 2, 6, 6, 9} et la valeur 10 pour MAX. On comptera comme correct un algorithme qui renvoie la valeur -3 ou la valeur 2 car ces deux valeurs apparaissent 4 fois chacune.

Plus particulièrement on demande le meilleur algorithme en termes d'ordre de complexité. La moitié des points des questions 1.2.1 à 1.2.3 est quand même obtenue si l'algorithme proposé est correct même s'il n'est pas le meilleur.

1.2.1) Indiquer en une à deux phrases le principe/l'ébauche de votre algorithme (0.5pt) :

1.2.2) donner le pseudocode ci-dessous (2 pt)

Element_le_plus_frequent
entrée : liste triée non-vide L d'entiers relatifs, taille de la liste N, entier MAX > 0 sortie : valeur entière signée la plus fréquente (cf donnée)

1.2.3) Justifier (avec une à deux phrases) l'ordre de complexité de votre algorithme en fonction de N avec la notation de Landau en O(…).(0.5pt)

1.2.4) On suppose maintenant que les valeurs de L restent comprises dans l'intervalle [-MAX, MAX] mais que la liste n'est pas triée. Est-ce possible de résoudre le problème de la question 1.2 avec un ordre de complexité meilleur que O(Nlog(N)) ? Justifiez votre réponse en trois à quatre phrases en indiquant ce que vous feriez pour cela. Si nécessaire, vous pouvez indiquer que vous utilisez des algorithmes mentionnés dans le cours. On ne demande pas de pseudocode, seulement le principe/l'ébauche de votre approche (1 pt)

Question 2: ***Type d'une matrice carrée A de taille N x N*** **(6 points)**

Soit une matrice carrée A non-vide de N lignes et de N colonnes contenant des nombres entiers. On accède à la valeur d'un élément de la matrice A avec la notation **A(i,j)** dans laquelle **i** est le **numéro de ligne** compris entre **1** et **N**, et **j** est le **numéro de colonne** compris entre **1** et **N**. Ecrire l'algorithme qui renvoie le type de la matrice parmi les 5 choix suivants :

- **Nulle** : Tous les éléments valent zéro.
- **TriangSup :** Triangulaire supérieure = tous les éléments ***au-dessous*** de la diagonale sont nuls
- **TriangInf** : Triangulaire inférieure = tous les éléments ***au-dessus*** de la diagonale sont nuls
- **Diagonale** : tous les éléments au-dessous et au-dessus de la diagonale sont nuls
- **Autre** : tous les cas différents des quatre cas précédents

Exemples :

3	5	8	9
0	1	7	2
0	0	5	1
0	0	0	6

L'algorithme renvoie **TriangSup**

3	0	0	0
0	1	0	0
0	0	5	0
0	0	0	6

L'algorithme renvoie **Diagonale**

3	5	0	0
4	1	0	0
0	0	5	2
0	0	7	6

L'algorithme renvoie **Autre**

2.1) Indiquer en deux à trois phrases le principe/l'ébauche de votre algorithme (1 pt) :

2.2) (***sur la page suivante***) proposez un algorithme pour résoudre ce problème (4pt)

2.3) Justifier (avec une à deux phrases) l'ordre de complexité de votre algorithme en fonction de N avec la notation de Landau en O(…). (1 pt)

typeMat
entrée : entier N>0, matrice d'entiers A non-vide, de N lignes x N colonnes sortie : un des 5 mot-clef parmi (Nulle, TriangSup, TriangInf, Diagonale, Autre)

Ne rien écrire sur cette page,

Rappel : avez-vous complété le tableau en p1 ?

Présenter cette page sur le dessus dans les 2 cas suivants :

1) *vous avez fini avant 15h45*
2) *les copies sont ramassées*