

ROYAUME DU MAROC
MINISTÈRE DE L'ENSEIGNEMENT
SUPÉRIEUR, DE LA RECHERCHE
SCIENTIFIQUE ET DE LA FORMATION
DES CADRES

Université Hassan II de Casablanca
Ecole Normale Supérieure
de l'Enseignement Technique Mohammedia



المملكة المغربية
وزارة التعليم العالي والبحث العلمي
وتكوين الأطر

جامعة الحسن الثاني بالدار البيضاء
المدرسة العليا لأساتذة التعليم التقني
المحمدية

Concours d'accès en première année du cycle d'ingénieurs Génie du Logiciel et des Systèmes Informatiques Distribués (GLSID)

Session : Juillet 2015

Epreuve de spécialité

Algorithmique et programmation

Durée : 3 heures

Remarques importantes :

- L'usage de la calculatrice ou de tout autre appareil électronique est interdit.
- Aucun document n'est autorisé.
- Les réponses à toutes les questions doivent être rédigées dans les feuilles de réponses.
- Chaque question du QCM ne peut avoir qu'une seule bonne réponse parmi les quatre propositions.
- Toutes les réponses au QCM doivent être reportées dans la feuille de réponse de la page 9 en plaçant pour chaque question une coche au centre de la case de la bonne colonne (A, B, C ou D).
- Les solutions algorithmiques peuvent être rédigées en utilisant un pseudo langage algorithmique ou l'un des langages de programmation suivants : C, C++, C#, Java.
- La clarté et la précision de vos solutions algorithmiques seront prises en considération.
- L'épreuve est notée sur un total de 100 points.

QCM : (20 points)

Reporter dans les feuilles de réponses, la lettre qui correspond à la bonne réponse :

1. Laquelle de ces étapes ne fait pas partie de la stratégie « diviser pour régner » ?
A) diviser le problème, C) résoudre les sous-problèmes,
B) trier les sous-problèmes, D) recombinaison des résultats.
2. Quel est le nombre d'appels à la fonction $f()$ suite à l'exécution du code suivant ?
`for(int i=0; i<n; i+=2) for(int j=1; j<n; f(), j*=2) ;`
A) $\theta(n)$, C) $\theta(n^2)$,
B) $\theta(n \log(n))$, D) $\theta(\log(n))$.
3. En plus d'un (ou plusieurs) appel à lui-même, que doit toujours comporter un algorithme récursif ?
A) une boucle for, C) une condition de terminaison,
B) une étape de division, D) une étape de fusion.
4. Quel est l'algorithme de tri qui est $O(n \log n)$ dans le cas moyen ?
A) Tri par sélection, C) Tri par monceau,
B) Tri à bulle, D) Tri par insertion.
5. Le tri fusion d'un tableau de n éléments se fait dans le meilleur cas en :
A) $\theta(n)$, C) $\theta(n^2)$,
B) $\theta(n \log(n))$, D) $\theta(\log(n))$.
6. Laquelle de ces structures de données est à utiliser pour implémenter efficacement une pile sans avoir à gérer les problèmes de dépassement de capacité (stack overflow) ?
A) un tableau, C) une liste chaînée,
B) un arbre binaire, D) un tas.
7. Laquelle de ces opérations ne s'effectue pas en temps constant sur une liste chaînée de n éléments ?
A) insertion en tête, C) suppression de l'élément en tête,
B) insertion à la position courante, D) accès à l'élément à la position $n/2$.
8. Partant d'une file vide, on effectue les opérations suivantes : `enfiler(3)`, `enfiler(7)`, `defiler()`, `enfiler(17)`, `defiler()`, `enfiler(11)`. Quelle valeur devrait renvoyer un nouvel appel à `defiler()` ?
A) 3, B) 7, C) 11, D) 17.

9. Pour quel programme parmi les suivants la pile serait la structure de données fondamentale ?

- A) Un correcteur orthographique,
- B) Un ordonnanceur de tâches,
- C) Un opérateur de complétion,
- D) Un analyseur syntaxique.

10. Pour un arbre binaire parfait de hauteur n , le nombre de nœuds est :

- A) 2^n ,
- B) $2^{n+1} - 1$,
- C) 2^{n+1} ,
- D) $n \log(n)$.

11. Soit un arbre binaire de recherche contenant des valeurs entières. On désire chercher la valeur

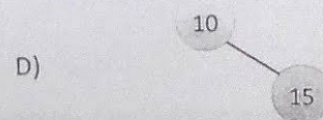
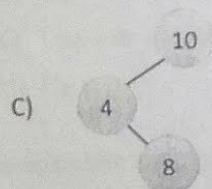
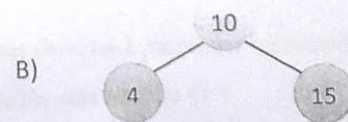
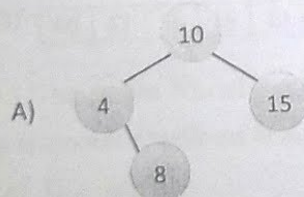
42. Quelle séquence parmi les suivantes, ne pourrait pas être une suite de nœuds parcourus ?

- A) 18 - 27 - 60 - 47 - 29 - 42,
- B) 57 - 18 - 53 - 55 - 48 - 42,
- C) 30 - 38 - 40 - 48 - 44 - 42,
- D) 36 - 46 - 38 - 44 - 40 - 42.

12. Pour supprimer un élément dans un arbre binaire de recherche :

- A) On le remplace par son fils gauche ou son fils droit,
- B) On le remplace par le plus petit élément de son fils gauche,
- C) On le remplace par le plus grand élément de son fils gauche, ou le plus petit élément de son fils droit, et on remplace celui-ci par son fils,
- D) On le remplace par le plus petit élément de son fils droit.

13. Parmi les arbres binaires de recherche suivants, lequel n'est pas un arbre AVL ?



14. Laquelle de ces structures de données permet de mieux implémenter une file de priorité ?

- A) Une liste,
- B) Un tas,
- C) Une table de hachage,
- D) Un graphe.

15. Pour insérer un élément dans un tas :

- A) on l'insère à la racine et on le fait descendre à sa place,
- B) on cherche sa place et on l'y insère en déplaçant le nœud présent,
- C) on l'insère à la fin du tas et on le fait remonter à sa place,
- D) on cherche une feuille plus grande que lui, et on l'insère comme son fils gauche.

16. Laquelle de ces opérations s'effectue toujours en temps constant dans une table de hachage ?

- A) Recherche,
- B) Suppression,
- C) Insertion,
- D) aucune des trois propositions.

17. On veut insérer l'élément $x=(cle, info)$ dans une table de hachage **tab** dont la fonction de hachage est **h**. La fonction **insérer(v, L)** insère l'élément **v** dans la liste **L**. Laquelle de ces opérations est la bonne ?

- A) $insérer(cle, tab[info])$,
- B) $insérer(h(x), tab[cle])$,
- C) $insérer(tab[cle], h(info))$,
- D) $insérer(x, tab[h(cle)])$.

18. Pour une implémentation standard de tas avec un tableau, lequel des tableaux suivants, ne représente pas un tas bien ordonné ?

- A)

55	24	18	14	11	16	10
----	----	----	----	----	----	----
- B)

66	17	50	16	10		
----	----	----	----	----	--	--

- C)

87	46	86	34	30	50	11
----	----	----	----	----	----	----
- D)

37	23	29	17	25	11	
----	----	----	----	----	----	--

19. Soit **G** un graphe orienté non valué, quasi complet, dont on cherche à parcourir les nœuds pour les énumérer. Quelle est la meilleure représentation possible des arcs de **G** ?

- A) Une matrice,
- B) Une table de listes chaînées,
- C) Une liste chaînée,
- D) Un tas.

20. On considère un graphe orienté possédant **n** arcs et **m** sommets. Quel espace mémoire occupe sa représentation par liste de successeurs ?

- A) $\theta(n)$,
- B) $\theta(m^2)$,
- C) $\theta(n + m)$,
- D) $\theta(n * m)$.

Exercice 1 : (12 points)

1. Ecrire un algorithme qui calcule le développement limité (DL) à l'ordre $2n+3$ de la fonction $\arctan(x)$ définie par :

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots + \frac{(-1)^n x^{2n+1}}{2n+1} + O(x^{2n+3})$$

2. Evaluer la complexité de votre solution.
3. Modifier l'algorithme de la question 1 pour obtenir un algorithme qui calcule la valeur approchée de π sachant l'expression suivante : $\frac{\pi}{4} = \arctan(1)$. L'algorithme doit s'arrêter lorsque l'erreur relative en valeur absolue est inférieure à une précision ε .
4. Sachant que le schéma de calcul basé sur le DL converge très lentement, quels risques encoure votre algorithme ? Proposer une alternative (sans la coder) pour éviter ces risques.

Exercice 2 : (18 points)

On cherche à étudier le problème de calcul de x^n , étant donné un réel x et un entier positif n . On souhaite faire ce calcul en se basant sur la règle suivante : Partant de $y_0 = x$, et si on a déjà calculé $y_1, y_2, \dots$ et y_{i-1} , on peut calculer y_i comme produit de deux résultats précédents arbitraires : $y_i = y_j y_k$, avec $0 \leq j, k < i$.

Le but est alors de calculer x^n le plus vite possible en minimisant i tel que $y_i = x^n$.

En appliquant la règle précédente, plusieurs schémas de calcul peuvent être trouvés. Parmi lesquels on distingue les deux schémas suivants :

- Schéma 1 : $y_i = y_0 y_{i-1}$
- Schéma 2 : $y_i = \begin{cases} y_{i/2} y_{i/2} & \text{si } i \text{ est pair} \\ x y_{\lfloor i/2 \rfloor} y_{\lfloor i/2 \rfloor} & \text{si } i \text{ est impair} \end{cases}$

Où $\lfloor i/2 \rfloor$ désigne la partie entière du rapport $i/2$.

1. Ecrire un algorithme itératif qui calcule x^n selon le schéma 1,
2. Ecrire un algorithme récursif qui calcule x^n selon le schéma 2,
3. Votre solution récursive représente-t-elle une récursivité terminale ? Justifier votre réponse,
4. Evaluer la complexité des deux algorithmes,

5. Donner la forme itérative équivalente à votre solution récursive,
6. Etant donné un ordinateur équipé d'un processeur 1GHz, donner une approximation du temps machine (en secondes) que fera cet ordinateur pour calculer x^n par les deux algorithmes quand ils sont appelés pour $x=1$ et $n=1e30$. On supposera que le processeur est utilisé à son entière capacité pour ce traitement et que l'opération de multiplication réelle coutera 10 cycles horloge. Pour simplifier les calculs, on fera l'approximation $3\log_2(10) \cong 10$. Quelle conclusion faites-vous en comparant les deux résultats ?

Exercice 3 : (10 points)

Soient $T1$ et $T2$ deux tableaux d'entiers ordonnés de tailles respectives $n1$ et $n2$. Ecrire un algorithme en $O(n)$ qui permet de trouver la Première Plus Longue Sous Chaîne Non Vide Commune à $T1$ et $T2$. Le résultat nommé $PPLSCNVC$, est un tableau de trois éléments représentant dans l'ordre l'indice de début de la sous chaîne dans $T1$, son indice de début dans $T2$ et sa longueur. Dans le cas où aucune sous chaîne non vide n'est trouvée, l'algorithme doit produire le résultat $PPLSCNVC = \{0, 0, 0\}$.

Exemple :

	0	1	2	3	4	5	6	7	8	9	10	11
$T1$	5	5	5	9	11	13	14	14	15	20	30	30

	0	1	2	3	4	5	6	7	8	9
$T2$	1	3	4	5	9	11	13	15	30	30

Le résultat doit être :

	0	1	2
$PPLSCNVC$	2	3	4

Exercice 4 : (20 points)

On considère une image couleur constituée des trois couches RGB. Les données de l'image sont stockées dans un tableau à trois dimensions : la première dimension correspond à la couleur (L'indice 0 pour la composante rouge, l'indice 1 pour la composante vert et l'indice 2 pour la composante bleu), la seconde correspond à la coordonnée selon $\vec{x}$ et la troisième à la coordonnée selon $\vec{y}$. Ainsi, les dimensions du tableau sont : $3 \times M \times N$ où M représente le nombre de pixels selon $\vec{x}$ et N le nombre de pixels selon $\vec{y}$. La valeur associée à chaque composante de couleur est un entier compris entre 0 et 255. La figure 1 du schéma suivant illustre la structure d'une image RGB.

Pour permettre facilement certains traitements sur l'image (classification, filtrage, reconnaissance, ...) des transformations préalables sont requises parmi lesquelles on s'intéresse aux suivantes :

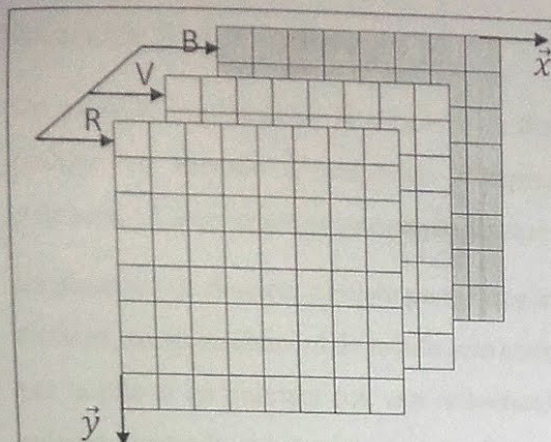


Figure 1 : Structure d'une image RGB

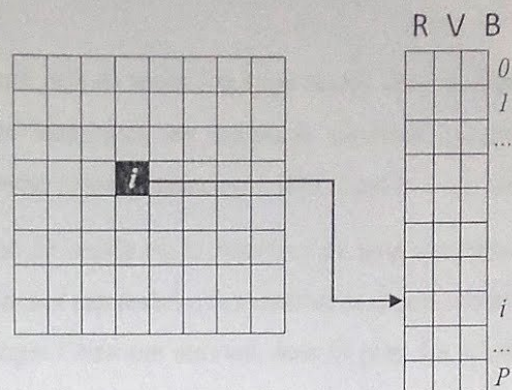


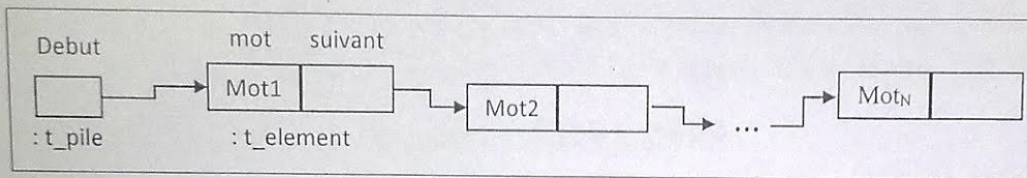
Figure 2 : Structure d'une image à palette de couleurs

- La transformation d'une image RGB de taille $3 \times M \times N$ en une image à palette de couleurs mémorisée sous forme d'une matrice de taille $M \times N$ d'entiers, chaque entier correspondant à un index dans une palette de couleurs qui est une matrice de taille $P \times 3$ où P désigne le nombre de couleurs distinctes contenues dans l'image. les trois colonnes de la matrice correspondent dans l'ordre aux composantes de rouge, de vert et de bleu (figure 2 du schéma ci-dessus).
 - La transformation d'une image RGB de taille $3 \times M \times N$ en une image en niveau de gris mémorisée sous forme d'une matrice de taille $M \times N$ contenant des valeurs entières comprises entre 0 et 255 (0 = noir à 255 = blanc). On adoptera la règle de calcul suivante pour calculer les niveaux de gris :
$$g = \frac{\alpha R + \beta V + \gamma B}{\alpha + \beta + \gamma}$$
 Où α, β et γ sont des entiers compris entre 0 et 255 tel que $\alpha + \beta + \gamma = 255$
 - Transformation d'une image niveau de gris de taille $M \times N$ en une image binaire mémorisée sous forme d'une matrice de taille $M \times N$ contenant des valeurs 0 et 1 (0 = noir à 1 = blanc). En fixant un seuil s entier compris entre 0 et 255, la valeur de l'image binaire transformée d'un niveau de gris g est définie à 0 (noir) si g est inférieure à s et à 1 (blanc) si non.
1. Proposer des types pour les trois formes d'images : *t_imgRGB*, *t_imgPalette*, *t_imgNvGris* et *t_imgBinaire*,
 2. Ecrire la fonction *imgRGB2imgPalette* qui reçoit une image RGB et qui retourne sa transformée en format palette de couleurs,
 3. Ecrire la fonction *imgPalette2imgRGB* qui reçoit une image en format palette de couleurs et qui retourne sa transformée en format RGB,
 4. Ecrire la fonction *imgRGB2imgNvGris* qui reçoit une image RGB et qui retourne sa transformée en format niveau de gris, On prendra pour $\alpha=54, \beta=182$ et $\gamma=19$,
 5. Ecrire la fonction *imgNvGris2imgBinaire* qui reçoit une image niveaux de gris et un seuil s et qui retourne sa transformée en format binaire.

Exercice 5 : (20 points)

On souhaite implémenter la structure de donnée d'une pile de mots (de type texte) dans le but de réaliser un module d'évaluation d'expressions arithmétiques en notation postfixe (notation polonaise). Cette structure permet de stocker des données selon le principe LIFO (Last In First Out).

La structure de données choisie pour l'implémentation de la pile est la structure de liste simplement chaînée. Ainsi un élément de la pile sera composé d'un mot (un texte) qui constitue la donnée stockée par la pile et un pointeur (ou une référence) qui indique l'élément suivant dans la pile. Le schéma suivant illustre la structure de donnée d'une pile sous forme de liste chaînée :



1. Donner la déclaration des types *t_element* et *t_pile*
2. Ecrire la fonction *estVide* qui permet de tester si une pile *P* est vide,
3. Ecrire la fonction *empiler* qui permet d'empiler un mot *m* dans une pile *P*
4. Ecrire la fonction *depiler* qui permet de dépiler le mot au sommet d'une pile *P*
5. On souhaite utiliser cette structure de pile pour permettre l'évaluation d'une expression arithmétique entière en notation postfixe fournie sous forme d'une chaîne de texte et dont les éléments (opérandes et opérateurs) sont obligatoirement séparés par au moins un espace.
 - a. Ecrire une fonction *decomposer* qui décompose le texte d'une expression arithmétique postfixe *E* en un tableau de mots (opérandes et opérateurs).

Exemple :

Si $E = "32 \ 24 \ + \ 3 \ 15 \ - \ *"$ le résultat produit par la fonction *decomposer* sera :

32	24	+	3	15	-	*
----	----	---	---	----	---	---

- b. Ecrire une fonction *getOperateur* qui reçoit un mot *m* et qui retourne le code opérateur correspondant. On convient de coder les opérateurs +, - et * respectivement par 1, 2 et 3. La fonction doit retourner 0 si le mot reçu ne correspond à aucun opérateur arithmétique.
- c. Ecrire une fonction *getNombre* qui reçoit un mot *m* et qui retourne le nombre correspondant lorsque *m* n'est composé que de chiffres. Si non une exception (erreur) doit être soulevée selon une notation à convenir.
- d. Ecrire une fonction *evaluerExpression* qui reçoit le texte d'une expression arithmétique postfixe *E* et qui retourne le résultat correspondant si l'expression est bien formée si non une erreur doit être soulevée selon une notation à convenir.