IN 101
12 mars 2012

Rattrapage

ENSTA ParisTech

NOM : **Prénom :**

- tous les documents (poly, slides, TDs, livres, brouillon du voisin...) sont **interdits**.
- ce QCM aboutit à une note sur 41 points. La note finale sur 20 sera obtenue simplement en divisant la note sur 42 par 2. Il suffit donc de donner 20 réponses justes (et aucune fausse) pour avoir la moyenne.
- n'oubliez pas de remplir votre nom et votre prénom juste au dessus de ce cadre.

```
1  int main(int argc, char* argv[]) {
2    int i;
3    int* tab;
4    ........
5    tab[0] = 2;
6    for (i=1; i<=5; i++) {
7      tab[i] = tab[i-1]*2 + 5;
8    }
9    printf("%d\n", tab[5]);
10   free(tab);
11   return 0;
12 }
```

1] Par laquelle des propositions suivantes faut-il remplacer la ligne `........` dans le code ci-dessus pour être certain qu'il s'exécute correctement ?

- ☐ `tab = (int*) malloc(5*sizeof(int*));`
- ■ `tab = (int*) malloc(6*sizeof(int));`
- ☐ `tab = (int) malloc(sizeof(6*int));`
- ☐ `tab = (int) calloc(5, sizeof(int));`

2] Qu'affiche le code ci-dessus quand on l'exécute (après avoir inséré la bonne ligne) ?

☐ 69 ☐ 107 ■ 219 ☐ 443

3] Laquelle des fonctions suivantes permet de calculer la même chose que le code ci-dessus (mais de façon récursive) en appelant `f(5)` ?

■
```
1 int f(int n) {
2   if (n==0) {
3     return 2;
4   }
5   return 2*f(n-1)+5;
6 }
```

☐
```
1 int f(int n) {
2   if (n!=0) {
3     return 2*(n-1)+5;
4   }
5   return 2;
6 }
```

☐
```
1 int f(int n) {
2   if (n>0) {
3     return f(2*(n-1)+5);
4   }
5   return 2;
6 }
```

☐
```
1 int f(int n) {
2   if (n<0) {
3     return 2;
4   }
5   return 2*f(n)+5;
6 }
```

4] Laquelle des boucles suivantes exécute la ligne `.....` exactement 10 fois ?

☐
```
for (i=0; i<=10; i++) {
  .....
}
```

☐
```
while (i<10) {
  i=0;
  .....
  i++;
}
```

■
```
i=0;
do {
  i++;
  .....
} while (i<10);
```

☐
```
for (i=0; i<10; i++) {
  .....
  i++;
}
```

5] Quels sont les types des arguments `argc` et `argv` que peut prendre la fonction `main` d'un programme en C ?

■ *int* et *char*** ☐ *int* et *int** ☐ *char* et *char** ☐ *int** et *char**

6] Laquelle des instructions suivantes est équivalente à l'instruction `a *= a;` ?

■ `a = a*a;` ☐ `a = *a;` ☐ `a = pow(a,a);` ☐ `a = a;`

7] Que retourne la fonction `malloc` quand elle n'arrive pas à allouer un bloc de la taille demandée (par exemple quand la mémoire est pleine) ?

☐ *void* ☐ -1
■ `NULL` ☐ on ne peut pas savoir

8] À quoi sert la fonction `free` en C ?

☐ à mettre un bloc mémoire à zéro,
☐ à vérifier si l'adresse mémoire passée en argument est utilisée par le programme,
■ à dire au système d'exploitation qu'un bloc mémoire ne sera plus utilisé (et peut donc être attribué à un autre programme),
☐ à rien.

9] Laquelle de ces complexités n'est pas polynomiale en n ?

☐ $\Theta(\log n)$ ☐ $\Theta(n \log n)$ ☐ $\Theta(2^{\log n})$ ■ $\Theta(2^{n \log n})$

10] Comment s'appelle un algorithme qui résout un problème en essayant de se rapprocher le plus possible de la solution à chaque étape ? Un exemple est l'algorithme de rendu de monnaie qui donne en premier le plus gros billet plus petit que le somme cible.

☐ un algorithme récursif, ☐ de la programmation dynamique,
■ un algorithme glouton, ☐ un algorithme NP-complet.

11] Quelle est la complexité en moyenne du tri d'un tableau de n éléments avec un algorithme de tri à bulles ?

☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ☐ $\Theta(n \log n)$ ■ $\Theta(n^2)$

12] Quelle est la complexité minimale d'un algorithme de tri par comparaison de n éléments ?

☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ■ $\Theta(n \log n)$ ☐ $\Theta(n^2)$

13] Lequel de ces algorithmes de tri fait toujours exactement le même nombre de comparaisons pour trier un tableau de 100 éléments, quel que soit leur ordre initial ?

☐ le tri rapide, ☐ le tri par tas,
■ le tri fusion, ☐ le tri par insertion.

14] Dans une implémentation normale (comme vue en cours) de *pile* ou de *file* contenant n éléments, quels sont les coûts respectifs des opérations d'insertion (fonction `push`) et d'extraction (fonction `pop`) d'un élément ?

☐ $\Theta(1)$ et $\Theta(n)$ ■ $\Theta(1)$ et $\Theta(1)$ ☐ $\Theta(n)$ et $\Theta(1)$ ☐ $\Theta(n)$ et $\Theta(n)$

15] Dans une implémentation normale (comme vue en cours) de *tas* contenant n éléments, quels sont les coûts respectifs des opérations d'insertion (fonction `push`) et d'extraction (fonction `pop`) d'un élément ?

☐ $\Theta(1)$ et $\Theta(\log n)$ ☐ $\Theta(\log n)$ et $\Theta(1)$
☐ $\Theta(1)$ et $\Theta(1)$ ■ $\Theta(\log n)$ et $\Theta(\log n)$

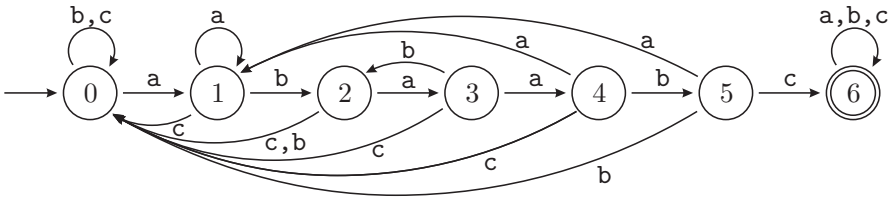
16] Dans une implémentation normale (comme vue en cours) de tas avec un tableau où le maximum est à la racine. Lequel des tableaux suivants correspond à un ordre acceptable ?

☐ | 42 | 18 | 27 | 15 | 21 | 23 | |
■ | 30 | 16 | 24 | 11 | 12 | 23 | 22 |

☐ | 47 | 39 | 20 | 34 | 17 | 19 | 30 |
☐ | 19 | 14 | 21 | 10 | 12 | 18 | |

17] On utilise un automate pour rechercher un motif de longueur m dans un texte de n caractères. L'alphabet est constitué de $|\Sigma|$ caractères différents. Si on néglige le temps de création de l'automate, quel est le coût de cette recherche ?

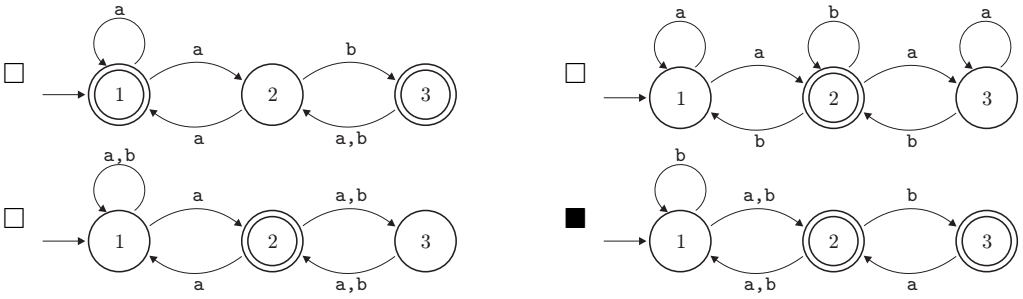
☐ $\Theta(m|\Sigma|)$ ■ $\Theta(n)$ ☐ $\Theta(nm)$ ☐ $\Theta(n \log n)$



18] L'automate déterministe ci-dessus permet la recherche du motif `abaabc` dans un texte, mais il contient une erreur ! De quel état la transition erronée part-elle ?

☐ 1 ☐ 3 ☐ 4 ■ 5

19] Lequel des automates non-déterministes suivants ne reconnaît pas le motif `abaa` ?



20] Si on représente l'expression arithmétique $4 \times (x+3)$ par un arbre, que contiendra la racine de l'arbre ?

☐ 4 ■ $\times$ ☐ $+$ ☐ x

21] Combien de feuilles un arbre binaire contenant 15 nœuds possède-t-il au minimum ?

■ 1 ☐ 2 ☐ 7 ☐ 8

22] Pour parcourir un arbre *en largeur*, il est nécessaire d'utiliser une structure annexe pour stocker les nœuds en attente de traitement. Quelle structure utilise-t-on en général pour cela ?

■ une file, ☐ une pile, ☐ un tas, ☐ un autre arbre.

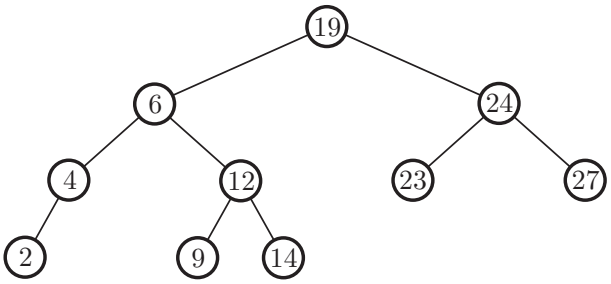


FIGURE 1 – Un arbre binaire de recherche.

23] Combien de *nœuds internes* l'arbre de la FIGURE 1 contient-il ?

☐ 4 ■ 5 ☐ 7 ☐ 10

24] Si l'on affiche le contenu de l'arbre de la FIGURE 1 à l'aide d'un parcours *préfixe*, dans quel ordre les nœuds seront-ils affichés ?

☐ 2,4,6,9,12,14,19,23,24,27
☐ 2,4,9,14,12,6,23,27,24,19
■ 19,6,4,2,12,9,14,24,23,27
☐ 19,6,24,4,12,23,27,2,9,14

25] Si l'on affiche le contenu de l'arbre de la FIGURE 1 à l'aide d'un parcours *postfixe*, dans quel ordre les nœuds seront-ils affichés ?

☐ 2,4,6,9,12,14,19,23,24,27
■ 2,4,9,14,12,6,23,27,24,19
☐ 19,6,4,2,12,9,14,24,23,27
☐ 19,6,24,4,12,23,27,2,9,14

26] Si l'on insère la valeur 17 dans l'arbre de la FIGURE 1, de quel nœud sera-t-il le fils ?

☐ 12 ■ 14 ☐ 19 ☐ 23

27] On considère maintenant que l'arbre de la FIGURE 1 est un arbre AVL. Si on insère alors la valeur 17 dans l'arbre, un déséquilibre va apparaître et il faudra faire une (ou des) rotations pour rétablir l'équilibre. À quel nœud de l'arbre ce déséquilibre apparaît-il ?

☐ 6 ☐ 12 ■ 19 ☐ 24

28] On considère toujours que l'arbre de la FIGURE 1 est un arbre AVL dans lequel on insère la valeur 17. Pour rétablir l'équilibre, quel type de rotation faut-il faire ?

☐ une rotation à gauche,
☐ une rotation à droite,
■ une double rotation à gauche puis droite,
☐ une double rotation à droite puis gauche.

29] Quelle est la complexité en moyenne d'une fonction `affiche(cell* L, int i)` qui affiche le i-ème élément de la liste chaînée `L` de longueur n qu'on lui passe en argument ?

☐ $\Theta(1)$ ☐ $\Theta(\log n)$ ☐ $\Theta(\sqrt{n})$ ■ $\Theta(n)$

30] Si on utilise la fonction `affiche(L,i)` de la question précédente dans une boucle `for` avec `i` variant de 0 à $n-1$ pour afficher la liste chaînée `L` en entier, quelle sera alors la complexité totale de l'affichage ?

☐ $\Theta(1)$ ☐ $\Theta(n)$ ☐ $\Theta(n \log n)$ ■ $\Theta(n^2)$

31] Que représente le *facteur de remplissage* d'une table de hachage ?

☐ le nombre de "cases" de la table de hachage,
■ le nombre moyen d'éléments dans chaque "case" de la table,
☐ le nombre total d'éléments dans la table,
☐ le nombre maximal d'éléments que l'on peut mettre dans une "case" de la table.

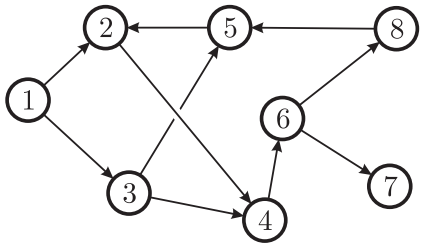


FIGURE 2 – Un graphe.

32] Laquelle de ces affirmations concernant le graphe de la FIGURE 2 est *fausse* ?

- ☐ c'est un graphe orienté,
- ■ c'est un graphe sans cycles,
- ☐ c'est un graphe connexe,
- ☐ ces trois affirmations sont vraies.

33] On effectue un parcours *en largeur* du graphe de la FIGURE 2 en partant du sommet 1. Cela nous donne un recouvrement du graphe qui permet de connaître :

- ☐ la longueur des cycles passant par le sommet 1,
- ☐ le nombre de chemins entre le sommet 1 et n'importe quel autre sommet,
- ■ le plus court chemin du sommet 1 vers n'importe quel autre sommet,
- ☐ le plus court chemin de n'importe quel sommet vers le sommet 1.

34] Si l'on écrit la matrice d'adjacence du graphe de la FIGURE 2, combien contiendra-t-elle de zéros et de uns ?

- ☐ 38 zéros et 18 uns
- ☐ 56 zéros et 8 uns
- ☐ 70 zéros et 11 uns
- ■ 54 zéros et 10 uns

35] L'algorithme de Dijkstra sert à chercher des plus courts chemins dans un graphe. Que permet-il de plus qu'un simple parcours ?

- ■ le graphe peut être pondéré,
- ☐ le graphe peut être non-orienté,
- ☐ le graphe peut contenir des cycles,
- ☐ le graphe n'a pas besoin d'être connexe.

36] On considère une fonction récursive de type diviser pour régner qui résout un problème de taille n en le divisant en deux sous-problèmes de taille $\frac{n}{2}$. Si le coût de la division/recombinaison est de $\Theta(n)$, quelle sera la complexité totale de la fonction ?

☐ $\Theta(n)$ ■ $\Theta(n \log n)$ ☐ $\Theta(n^2)$ ☐ $\Theta(2^n)$

37] On considère une fonction récursive de type diviser pour régner qui résout un problème de taille n en le divisant en deux sous-problèmes de taille $\frac{n}{2}$. Si le coût de la division/recombinaison est de $\Theta(n^2)$, quelle sera la complexité totale de la fonction ?

■ $\Theta(n^2)$ ☐ $\Theta(n^2 \log n)$ ☐ $\Theta(n^3)$ ☐ $\Theta(2^n)$

```
1 void f(type* V) {
2   if (V != NULL) {
3     printf("%d\n",V->val);
4     f(V->next);
5   }
6 }
```

38] La fonction `f` ci-dessus permet d'afficher le contenu de quel type de structure de données ?

- ☐ un tas,
- ☐ un arbre général,
- ■ une liste chaînée,
- ☐ une chaîne de caractères.

39] Que ce passe-t-il si l'on inverse les lignes 3 et 4 du code de la fonction `f` ci-dessus ?

- ☐ la fonction fera une boucle infinie,
- ■ l'affichage se fera en sens inverse,
- ☐ la fonction n'affichera plus rien,
- ☐ rien ne sera changé.

40] La fonction `f` ci-dessus peut être qualifiée de fonction *récursive terminale*. Pourquoi ?

- ☐ elle ne comporte pas de `return`,
- ■ l'appel récursif est toujours la dernière instruction exécutée,
- ☐ elle ne comporte qu'un seul appel récursif,
- ☐ elle possède une condition d'arrêt.

41] À quoi sert la commande `break` en C ?

- ☐ à quitter une fonction,
- ☐ à renvoyer une erreur,
- ■ à interrompre une boucle,
- ☐ à faire un test.