Nom :
Prénom :

L2 Algorithmique des arbres

Durée 2 heures.

Notes de cours, de TD et de TP autorisées.

- Les différentes parties sont indépendantes et peuvent être traitées dans n'importe quel ordre.
- On veillera à minimiser la complexité des fonctions.
- On pourra écrire des fonctions annexes, si l'utilisation de paramètres supplémentaires est nécessaire.
- Le présent sujet est à rendre avec sa copie (voir Exercice 3)

▸ Exercice 1. Arbres binaires

On utilise la structure :

```
typedef struct noeud{
    int val;
    struct noeud *fg, *fd;
} Noeud, *Arbre;
```

1. Écrire une fonction `int nombreFilsUniques(Arbre a)` qui renvoie le nombre de nœuds internes ayant un unique fils.
2. Écrire une fonction `int estTournoi(Arbre a)` qui renvoie 1 si `a` est un arbre tournoi et 0 sinon.
   Rappel : Un arbre est dit tournoi lorsque l' étiquette d'un nœud est supérieure ou égale à l'étiquette de son père.
3. Écrire une fonction `int plusCourte(Arbre a)` qui renvoie la hauteur de la plus courte branche.

▸ Exercice 2. Arbre lexicographique

On représente un arbre lexicographique à l'aide d'un arbre ternaire de recherche

```
typedef struct neuLex{
    char lettre;
    struct neuLex *frg,*fils,*frd;
} NoeudT, * ArbreT;
```

1. Ecrire une fonction `int nombreMots(ArbreT a)` qui renvoie le nombre de mots mémorisés par l'arbre.

2. Ecrire une fonction `int estPrefixe(ArbreT a, char mot[])` qui renvoie 1 si `mot` est le préfixe, au sens large, d'un mot mémorisé dans l'arbre.

▸ Exercice 3. Compression de Huffman

On a reçu un fichier compressé par la méthode de Huffman contenant :

0 1 1 0 0 0 1 [H] 1 [O] 0 1 [P] 0 1 [!] 1 [A] 0 0 0 1 [D] 1 [E] 0 1 [R] 1 [W]
0 1 [L] 0 1 [Y] 1 [esp] 0 0 0 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0
1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 0 1

Ici, `esp` désigne le caractère espace.

1. Dessiner l'arbre de Huffman associée à ce codage, en expliquant la méthode de construction.

2. Donner le codage utilisé pour chaque lettre dans le tableau ci-dessous :

Lettre	Codage associé
!	
espace	
A	
D	
E	
H	

Lettre	Codage associé
L	
O	
P	
R	
W	
Y	

3. Décompresser le texte, en expliquant la méthode utilisée.

▸ Exercice 4. Tas

1. Le tableau suivant représente t-il un tas ?

2	5	3	11	18	6	16	10	12

   Si oui, dessiner le tas correspondant sous forme d'arbre. Si non, faire les échanges nécessaires pour qu'il devienne un tas en dessinant clairement les échanges effectués.

2. Donner le tableau obtenu après la suppression du minimum du tas précédent. On donnera tous les échanges intermédiaires nécessaires pour maintenir la structure de tas en dessinant les arbres intermédiaires correspondants.

▸ Exercice 5. AVL

On ajoute successivement dans un arbre vide les valeurs : 7, 3, 2, 4, 1, 6, 5. Puis, on supprime 3.
Dessiner les arbres obtenus après chaque ajout ou suppression, ainsi que les arbres intermédiaires s'il y a rotation. Dans ce cas,on indiquera aussi clairement quelle rotation est utilisé et

sur quel nœud elle s'applique. Pour les suppressions, on fera remonter le plus grand élément inférieur à celui qui est supprimé ; si cela n'est pas possible, on fera alors remonter le plus petit élément supérieur.

▸ Exercice 6. Ensembles, représentation par tableau des pères
Pour représenter un ensemble d'arbres, on utilise la structure suivante :

```
typedef struct {
    int *peres;
    int nbNoeuds;
} Foret;
```

Les nœuds des arbres seront les entiers 0, 1, 2, ..., `nbNoeuds`−1. Aucun de ces entiers ne sera utilisé deux fois.

Si `f` est une `Foret`, `f.peres[i]` est le numéro du père du nœud `i`. On considère que la racine d'un arbre est son propre père. On supposera que la forêt est bien définie ; on ne cherchera pas à le vérifier.

1. Écrire une fonction `int nbArbres(Foret f)` qui renvoie le nombre d'arbres présent dans la forêt `f`.

2. Écrire une fonction `int plusGrand(Foret f)` qui renvoie le numéro de la racine de l'arbre ayant le plus grand nombre de nœuds. En cas d'égalité de nombre de nœuds maximum, peu importe quel arbre sera désigné par la fonction `plusGrand`.
   L'algorithme utilisé devra être explicitement expliqué. Il sera de complexité linéaire.

+1/1/60+

Examen algorithmique des arbres **Partie Q.C.M.**

Université Paris-Est Marne-la-Vallée

• Vous devrez donner vos réponses à ce Q.C.M. en noircissant (= au stylo noir) la ou les cases correspondantes aux réponses choisies.

• Les questions marquées d'un trèfle peuvent avoir plusieurs réponses correctes. Il faut alors noircir plusieurs cases.

• Des réponses fausses peuvent entrainer des points négatifs.

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9
☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9

⟵ Codez votre numéro d'étudiant ci-contre de la manière suivante : un chiffre par ligne, le premier sur la première ligne, etc. Écrivez votre nom et prénom ci-dessous.

Nom et prénom :

Question 1 L'ensemble de mots binaires $\{001, 0011, 01, 011\}$ est-il un code ?

☐ oui ☐ non

Question 2 Combien de feuilles, au maximum, un tas de hauteur 8 contient-il ?

☐ 256 ☐ 129 ☐ 128 ☐ 257

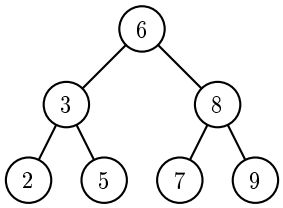
Question 3 ♣ On considère la fonction suivante:

```
void faitUnTruc(Arbre a, int n){
  if(a == NULL)
     return;
  a -> val = n;
}
```

Soit `a` un arbre binaire dont le parcours préfixe est `1 2 3 4`. Après l'appel de fonction `faitUnTruc(a, 5)`, quel est l'affichage préfixe de `a`?

☐ `5 2 3 4` ☐ `5 1 2 3 4` ☐ `1 2 3 4` ☐ On ne sait pas

Question 4 ♣ L'arbre suivant est un ...



☐ AVL ? ☐ B-arbre de degré 2 ? ☐ ABR ? ☐ B-arbre de degré 3 ?

Question 5 Quelle est la hauteur maximum d'un arbre avl contenant 42 nœuds ?

☐ 7 ☐ 41 ☐ 4 ☐ Cela dépend de l'ordre des ajouts

Question 6 On considère un tas contenant n éléments. Quelle est la complexité minimale d'un algorithme permettant d'extraire le plus grand élément du tas ?

☐ $\mathcal{O}(n\log(n))$ ☐ $\mathcal{O}(n)$ ☐ $\mathcal{O}(\log(n))$ ☐ $\mathcal{O}(1)$

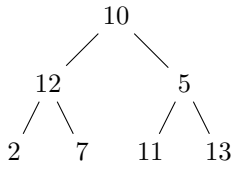
Question 7 Un texte contient uniquement les lettres {a,b,c,d,e,f} réparties comme suit :

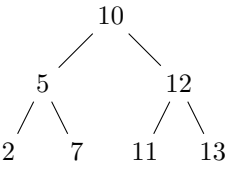
- a apparait 7 fois
- b apparait 3 fois
- c apparait 3 fois
- d apparait 12 fois
- e apparait 4 fois
- f apparait 5 fois

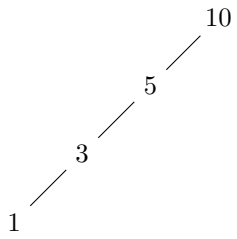
Quel est la taille en bits du texte compressé seul (sans codage de l'arbre)

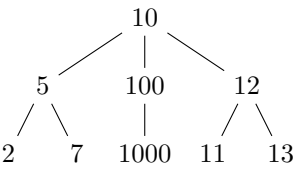
☐ 89 bits
☐ 83 bits
☐ cela dépend de la construction de l'arbre
☐ 68 bits
☐ Aucune de ces réponses

Question 8 ♣ Parmi les arbres suivants, lesquels sont des arbres binaires ?

☐ 

☐ 

☐ 

☐ 

Question 9 Un ensemble de parties de l'ensemble $\{0, 1, \ldots, 9\}$est représenté par le tableau de pères

1	3	1	3	0	3	1	7	9	9

Quel est le tableau `père` après la fusion de l'ensemble contenant 7 et de celui contenant 4 en appliquant la compression du chemin et la fusion par rang

☐

1	3	1	7	3	5	1	7	9	9

☐

1	3	1	3	0	7	1	3	9	9

☐ Aucun de ces tableaux

☐

3	3	1	3	3	3	1	3	9	9

☐

3	3	1	3	0	7	1	7	9	9

Question 10 Un fichier contient :

```
110001[a]01[c]1[b]01[d]1[e]10110100001100101000
```

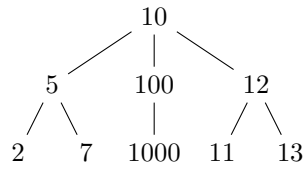
Quel texte a été compressé par la méthode de Huffman ?

☐ dabeceba
☐ aucun de ces textes
☐ decabadda
☐ decaba

Question 11 ♣ Soit `a` un arbre binaire dont l'affichage d'un parcours suffixe est `5 4 3 2 1` et dont l'affichage d'un parcours infixe est `1 5 2 4 3`. Quel est l'affichage préfixe de `a`?

☐ Aucune des réponses ☐ `1 5 2 4 3` ☐ `5 4 3 2 1` ☐ `1 2 5 3 4`

Question 12 ♣ On considère l'arbre suivant :



On effectue un parcours de l'arbre et on obtient :

```
...  1000 12 11 ...
```

Quel parcours avons nous réalisé ?

- ☐ cette suite d'entiers ne correspond pas à un parcours de cet arbre.
- ☐ un parcours en largeur
- ☐ un parcours en profondeur infixe
- ☐ un parcours en profondeur suffixe
- ☐ un parcours en profondeur préfixe

Question 13 ♣ Parmi les ensembles ci-dessous, lesquels peuvent être obtenus comme des codes de Huffman ?

☐ $\{0, 01, 011, 111\}$ ☐ $\{0, 10, 110, 1110\}$ ☐ $\{00, 01, 10, 11\}$
☐ $\{0, 10, 110, 111\}$

Question 14 L'arbre de Huffman en cours de construction est contenu dans le tableau ci-dessous :

a b c d e	1 2 2 2 5		
	3 4	0 2	1 3

Compléter les lignes manquantes dans ce tableau.
NE PAS COCHER LES CASES

réservé à la note : ☐faux ☐approximatif ☐moyen ☐bien ☐parfait