Algorithmique 1

DS 2

Documents **non** autorisés

NB : Dans toutes les fonctions qui sont demandées, il faudra utiliser uniquement les primitives (aucune référence à la structure). Ces primitives sont rappelées en annexe

Exercice 2.1

Un palindrome est une chaîne de caractères qui se lit de la même manière de gauche à droite et de droite à gauche. En utilisant une seule pile, une seule files, et une seule variable du type `int`, écrire un algorithme qui détermine si une chaîne de caractère est un palindrome. On suppose que la chaîne de caractères est passée en paramètre dans une liste chaîne de type `listeSC` qui ne devra être parcourue qu'une seule fois. L'algorithme doit donner en sortie la longueur de la chaîne si c'est un palindrome sinon 0.

Exercice 2.2

Dans l'exercice suivant, on considère l'algorithme récursif de parcours type `arbreBinaire` vu en cours

```
fonction parcoursArbreBinaire(val A:arbreBinaire d'objet):vide;
1.    Declarations locales
2.    debut
3.    traitement1;
4.      si estFeuille(A)alors
5.      traitement2
6.      sinon
7.      traitement3;
8.         si filsGauche(A)!=NIL alors
9.        traitement4;
10.          parcoursArbreBinaire(filsGauche(A));
11.        traitement5;
12.        finsi
13.      traitement6;
14.        si filsDroit(A)!=NIL alors
15.        traitement7
16.          parcoursArbreBinaire(filsDroit(A));
17.       traitement8;
18.        finsi
19      traitement9;
20      finsi
21    traitement10;
22   fin
```

Indiquer les numeros des instructions correspondants à l'affichage des valeurs des sommets de l'arbre

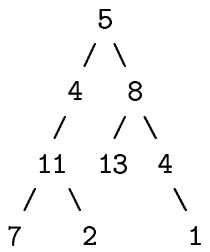
1. dans l'ordre prefixe
2. dans l'ordre infixe
3. dans l'ordre postfixe (suffixe)

En d'autre terme $\texttt{traitement}_i \equiv$ `print (getValeur(S))` ;
Justifier vos réponses.

Exercice 2.3

Ecrire un algorithme qui calcule la longueur des cheminements externes (autrement dit : on ne considère que les feuilles).

Exemple



- path 1 : 5 4 11 7
- path 2 : 5 4 11 2
- path 3 : 5 8 13
- path 4 : 5 8 4 1

Pour l'arbre ci-dessus, la fonction renvoie comme résultat : $11 = (3 + 3 + 2 + 3)$

Exercice 2.4

Ecrire une fonction qui calcule la moyenne de valeurs de feuilles d'un arbre binaire d'entier, en faisant un seul parcours de l'arbre (En d'autres termes, chaque sommet sera visité une et une seule fois. Toute autre solution sera refusée.)

ANNEXE

```
arbreBinaire=curseur;
sommet=curseur;
```

Le type sommet présente les primitives suivantes :

- accès

```
fonction getValeur(val S:sommet):objet;
/* vaut NIL si le sommet n'existe pas /*
fonction filsGauche(val S:sommet):sommet;
/* vaut NIL si S n'a pas de fils gauche */
fonction filsDroit(val S:sommet):sommet;
/* vaut NIL si S n'a pas de fils droit */
fonction pere(val S:sommet):sommet;
/* vaut NIL si S est la racine de l'arbre */
```

- modification

```
fonction setValeur(ref S:sommet;val x:objet):vide;
/* affecte au sommet S la valeur x */
fonction ajouterFilsGauche(ref S:sommet,val x:objet):vide;
/* filsGauche(S)==NIL doit être vérifié */
fonction ajouterFilsDroit(ref S:sommet,x:objet):vide;
/* filsDroit(S)==NIL doit être vérifié */
fonction supprimerFilsGauche(ref S:sommet):vide;
/* filsGauche(S) est une feuille */
fonction supprimerFilsDroit(ref S:sommet):vide;
/* filsDroit(S) est une feuille */
fonction detruireSommet(ref S:sommet):vide;
/* S est une feuille */
```

- Il faut par ailleurs pouvoir créer la racine d'un arbre on a donc de plus la primitive

```
fonction creerArbreBinaire(val Racine:objet):sommet;
```

Pour rappel les primitives de pile et file

```
fonction valeur(ref P:pile de objet):objet;
fonction pileVide(ref P:pile de objet):bool~en;
fonction cr~erPile(P:pile de objet);
fonction empiler(ref P:pile de objet;
                 val x:objet):vide;
fonction d~piler (ref P:pile de objet):vide;
fonction detruirePile(P:pile de objet):vide;

fonction valeur(ref F:file de objet):objet;
fonction fileVide(ref F:file de objet):bool~en;
fonction cr~erFile(F:file de objet);
fonction enfiler(ref F:file de objet;
                 val v:objet):vide;
fonction d~filer(ref F:file de objet):vide;
fonction detruireFile(F:file de objet):vide;
```