Algorithmique 1

DS 3

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

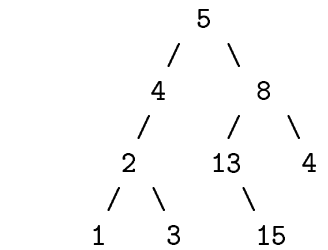
Exercice 3.1

Soit la fonction mystere suivante :

```
fonction mystere(val T: arbreBinaire d'entier): booleen;
var min , max : entier;
debut
      min = INT_MIN ; // le plus petit entier
      max = INT_MAX ; // le plus grand entier
      retourne mystere-aux( T, min, max);
fin

fonction mystere-aux(val T :arbreBinaire d'entier,val min: entier,val
 max: entier): booleen;
debut
     si  (T==NULL) alors  retourne (vrai); fsi
     si  (getValeur(T) < min  ou getValeur(T) > max) alors retourne (faux);
     sinon
           retourne
               ( mystere-aux( filsGauche(T), min, getValeur(T)) et
                 mystere-aux( filsDroit(T), getValeur(T)+1, max));
     fsi
fin
```

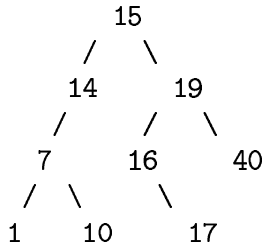
1. Faites la trace de l'appel de `mystere` sur l'arbre binaire ci-dessous.



2. Que fait `mystere` en général ?
3. Que faut-il modifier pour que `mystere` puisse être appliqué à un arbre planaire

Exercice 3.2

Ecrire une fonction qui détermine le k-ième élément d'un arbre binaire de recherche sans utiliser une structure de données supplémentaire.
Exemple



Sur l'arbre binaire de recherche ci-dessus, l'appel à `k-ieme-element(T,6)` renvoie 16 et celui de `k-ieme-element(T,10)` renvoie 0.

Exercice 3.3

A l'aide de la fonction `k-ieme-element(T,k)` Ecrire une fonction qui détermine l'élément médian d'un arbre binaire de recherche.

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;
fonction filsDroit(val S:sommet):sommet;
fonction pere(val S:sommet):sommet;
```

- 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;
```