Option Informatique MPSI, DS n°1

Les questions de code devront être traitées en language OCaml. Les noms de variables doivent être suffisamment clairs, et le code compréhensible.

Si vous définissez une fonction auxiliaire, précisez le type des arguments et l'objectif de la fonction.

Les questions difficiles sont notées d'une étoile *. N'hésitez pas à les sauter.

I) Autour des polynômes

Dans cette partie, on va étudier une représentation des polynômes d'entiers en OCaml.

On prend un polynôme P :

$$P(X) = \sum_{k=0}^{n} a_k X^k \quad \text{pour} \quad a_0, ..., a_n \in \mathbb{N} \tag{1}$$

On représente les polynômes en OCaml par la liste contenant $a_0, ..., a_n$. On se donne donc le type suivant :

```
type polynome = int list;;
```

Par exemple, le polynôme $X^3 + 1$ sera représenté par la valeur OCaml suivante :

```
let cube = [1; 0; 0; 1];;
```

Q1) Écrire une fonction `degres: polynomes -> int` qui renvoie le degrés d'un polynôme (c'est-à-dire n).

Q2) Écrire une fonction `dominant: polynomes -> int` qui renvoie le coefficient dominant d'un polynôme (c'est-à-dire a_n).

Q3) Écrire une fonction `xn: int -> polynome` qui prend en argument un entier n et renvoie le polynôme X^n

Q4) Écrire une fonction `mul_scalaire: polynomes -> int -> polynome` qui multiplie un polynôme par un entier.

Q5) Écrire une fonction `add: polynomes -> polynome -> polynome` qui réalise la somme de deux polynômes.

```
let rec eval (p: polynome) (x: int): int = match p with
    |[] -> 0
    |a::q -> a + x * (eval q x);;
```

Q6) Que fait la fonction `eval` ? Quelle est sa complexité ?

Q7) * Écrire une fonction `mul: polynomes -> polynome -> polynome` qui multiplie deux polynômes.

II) Arbres binaires de recherche

II.1) Généralités

On étudie des arbres binaires de recherche, définis par induction de la manière suivante :

Définition 1 [Arbre binaire de recherche]

> Un arbre binaire de recherche est :
> - Soit une feuille F ne stockant rien
> - Soit un noeud interne N stockant un entier x, et deux sous arbres G et D, tels que :
>   - G et D sont des arbres binaires de recherche
>   - G stocke uniquement des éléments plus petits que x, et D stocke uniquement des éléments plus grands que x (strictement).

On utilise le type OCaml suivant pour stocker les arbres binaires de recherche :

```
type abr =
    | F
    | N of abr * int * abr;;
```

Q8) Écrire une fonction `hauteur: abr -> int` qui calcule la hauteur d'un arbre binaire de recherche.

Q9) Écrire une fonction `min: abr -> int` qui renvoie la valeur minimum stockée.

Q10) Écrire une fonction `recherche: abr -> int -> bool` qui prend en argument un arbre et un entier, et renvoie un booléen indiquant si l'entier est présent dans l'arbre. Quelle est sa complexité en fonction de h la hauteur de l'arbre ?

Q11) Expliquer comment on peut procéder pour insérer un élément dans l'arbre, et avec quelle complexité en fonction de h.

Q12) De même pour supprimer un élément.

Q13) Soit un arbre de taille n. Indiquer le meilleur et le pire cas pour la complexité des opérations en fonction de n, par rapport à la forme de l'arbre.

II.2) Arbres Bicolores

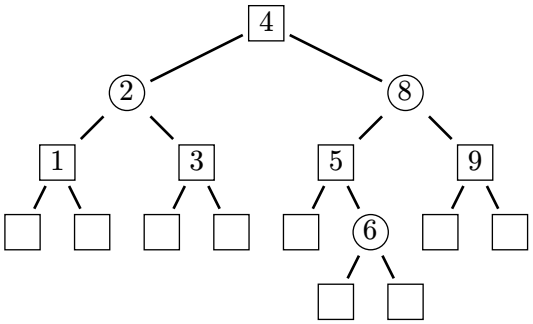
On va introduire une méthode pour garder les arbres équilibrés, et donc toujours avoir une bonne complexité pour nos opérations. Habituellement on utilise des couleur, mais pour des raisons de photocopie en noir et blanc, on utilisera des formes ici.

Définition 2 [Arbre bicolore]

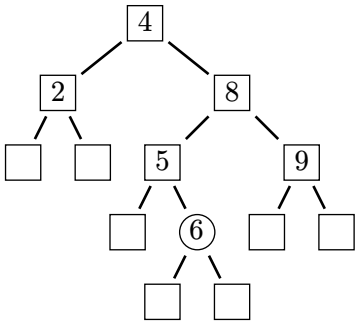
> Un arbre bicolore est un arbre binaire de recherche dans lequel on attribue une forme (carré ou rond) à chaque noeud, avec les contraintes suivantes :
>
> - La racine est **carré**
> - Les enfant d'un noeud **rond** sont **carré**
> - Tous les chemins de la racine à une feuille ont le même nombre de noeuds **carré**
> - Les feuilles sont **carré**

Q14) Les arbres suivants sont-ils des arbres bicolores ?

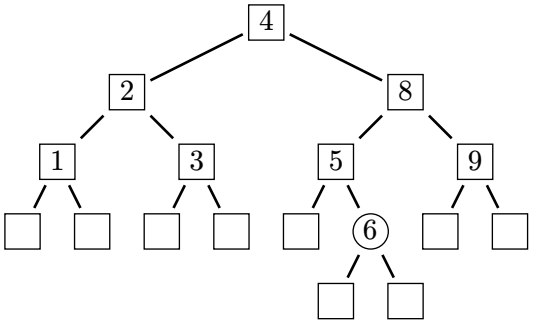
(a)



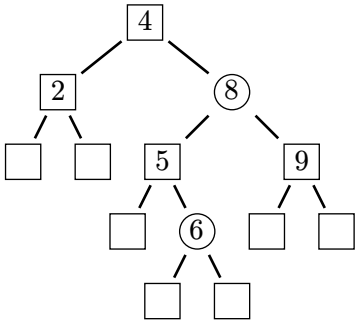
(b)



(c)



(d)



Q15) Quelle est la hauteur maximale possible pour un arbre bicolore de taille 5 ? Et de taille 17 ?

Q16) * Généralisez avec un arbre de taille $2^n + 1$, et démontrez votre résultat.

Q17) En déduire une borne sur la hauteur d'un arbre bicolore en fonction de sa taille.

Q18) En supposant que l'on puisse maintenir la propriété bicolore lors des opérations de recherche, ajout et suppression, sans coût supplémentaire, quelle serait alors la complexité dans le pire des cas de ces opérations ?