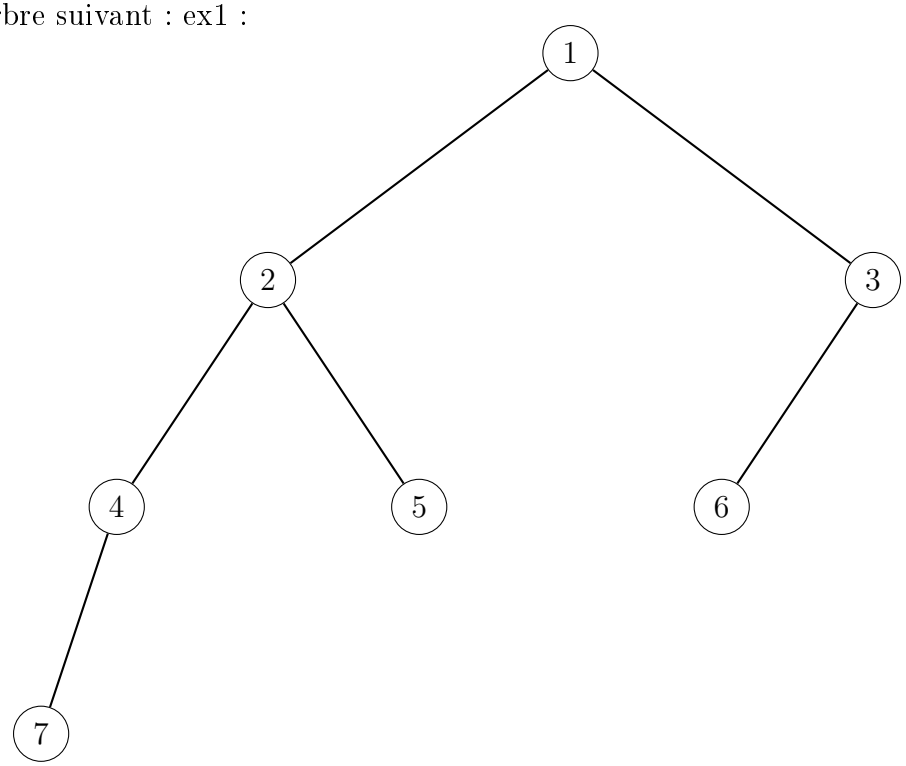
TD 8

Arbres binaires

On utilisera le type suivant :

```
type 'a arbre_bin =
  | Vide
  | Noeud of  ('a arbre_bin)*'a*('a arbre_bin)
;;
```

1. Définir l'arbre suivant : ex1 :



2. Écrire une fonction `taille` qui calcule la taille d'un arbre.

   `taille : 'a arbre_bin -> int`

3. Écrire une fonction `hauteur` qui calcule la hauteur d'un arbre :

   `hauteur : 'a arbre_bin -> int`

4. Écrire une fonction `maxi` qui renvoie la plus grande étiquette d'un arbre.

   `maxi : 'a arbre_bin -> 'a`

5. Écrire une fonction `test_complet` qui vérifie si un arbre est complet.

   `test_complet : 'a arbre_bin -> bool`

6. On souhaite construire un arbre binaire équilibré à partir d'un tableau. On propose de prendre pour racine l'élément d'indice 0 du tableau, puis pour un nœud étiqueté par un élément d'indice k, le fils gauche est l'élément d'indice $2k+1$, et le fils droit l'indice $2k+2$, sous réserve d'existence. Écrire la fonction `arbre_of_vect` qui convertit un tableau en arbre binaire équilibré.

   `arbre_of_vect : 'a array -> 'a arbre_bin`

Parcours d'un arbre

7. Parcours en pré-ordre (préfixe) :
   Définition : Parcourir un arbre binaire non vide en pré-ordre consiste à traiter la racine de cet arbre, puis à parcourir en pré-ordre l'arbre fils gauche de cette racine et enfin, à parcourir en pré-ordre l'arbre fils droit de cette racine. (parcourir un arbre binaire vide en pré-ordre est une opération nulle (pas de traitement)).

   Écrire une fonction `prefixe` qui prend en argument un arbre et rend la liste de ses arguments, donnée en parcourant l'arbre de façon préfixe.

   ```
   prefixe : 'a arbre_bin -> 'a list
   ```

8. **Parcours en ordre (infixe) :**

   Définition : Parcourir un arbre binaire non vide en ordre (ou de façon infixe) consiste à parcourir en ordre l'arbre fils gauche de cet arbre, puis à traiter la racine, et enfin, à parcourir en ordre l'arbre fils droit de cette racine. (parcourir un arbre binaire vide en pré-ordre est une opération nulle (pas de traitement)).

   Écrire une fonction `infixe` qui prend en argument un arbre et rend la liste de ses arguments, donnée en parcourant l'arbre de façon infixe.

   ```
   infixe : 'a arbre_bin -> 'a list
   ```

9. **Parcours en post-ordre (postfixe) :**
   Définition : Parcourir un arbre binaire non vide en post-ordre (ou de façon postfixe) consiste à parcourir en post-ordre l'arbre fils gauche de cette arbre, puis à parcourir en post-ordre l'arbre fils droit, et enfin à traiter la racine. (parcourir un arbre binaire vide en pré-ordre est une opération nulle (pas de traitement)).

   Écrire une fonction `postfixe` qui prend en argument un arbre et rend la liste de ses arguments, donnée en parcourant l'arbre de façon infixe.

   ```
   postfixe : 'a arbre_bin -> 'a list
   ```

10. **Parcours militaire :**
    Il faut ajouter l'ordre militaire, qui consiste à lire profondeur par profondeur, de gauche à droite. C'est un parcours en largeur d'abord.

    Écrire une fonction `militaire` qui prend en argument un arbre et rend la liste de ses arguments, donnée en parcourant l'arbre de façon militaire.

    ```
    militaire : 'a arbre_bin -> 'a list
    ```