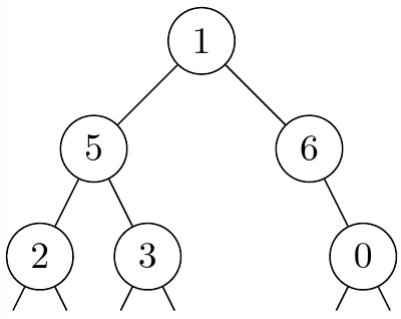
TD n°13 – Arbres

Dans ce TD on se place en C et on considère les types suivants pour les arbres binaires et les arbres enracinés :

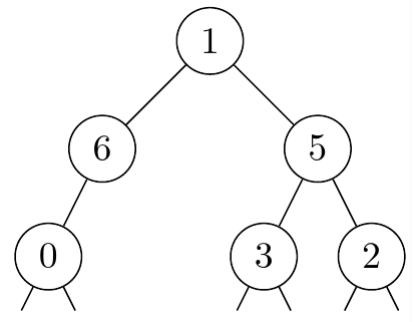
```c
struct Arbrebin{
 int etiquette;
 struct Arbrebin* g;
 struct Arbrebin* d;
};

typedef struct Arbrebin arbrebin;

struct Arbre{
 int etiquette;
 struct Arbre** enfants;
 int arite;
};

typedef struct Arbre arbre;
```

1 Arbres binaires

1. Écrire en C une fonction calculant la hauteur d'un arbre binaire. La signature sera `int hauteur (arbrebin* a)`.
2. Écrire en C une fonction `int dernier(bintree* a)` qui renvoie l'étiquette du noeud le plus à droite de l'arbre. (le noeud le plus à droite est le noeud qu'on atteint en allant que à droite)
3. Écrire en C une fonction `void echange_sous_arbre(bintree* a)` qui échange le sous-arbre gauche de la racine et le sous-arbre droit de la racine.
4. Écrire en C une fonction `void miroir(bintree* a)` qui transforme l'arbre en entrée en son miroir (sa symétrie axiale verticale).
   Par exemple le miroir de :



est



2 Arbres enracinés

5. Écrire une fonction qui réalise un parcours en profondeur préfixe d'un arbre non-binaire, en C. Le traitement consistera en l'affichage des noeuds.
6. Écrire une fonction qui réalise un parcours en largeur d'un arbre non-binaire, en C.
   On pourra supposer l'existence d'un type `file*` qui implémente une file (avec les noms de primitives habituels).