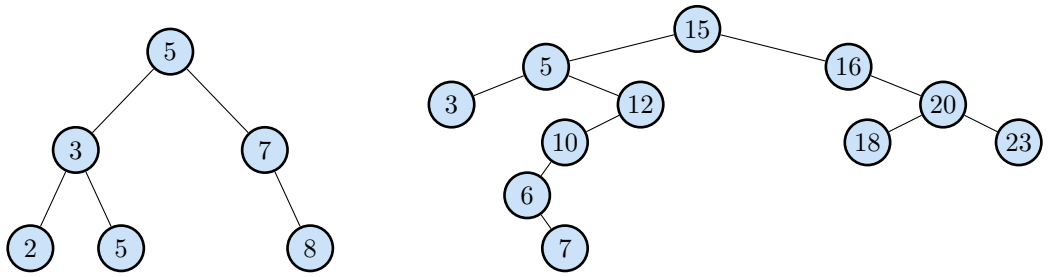
TP n°18 - Arbres binaires en C.

Toutes les fonctions de ce TP seront codées dans un script OCaml `NOM_bintree.c`.

Nous allons implémenter aujourd'hui en C la structure de données abstraite d'arbre binaire que nous avons commencé à étudier la semaine dernière.

Exercice 1 (Type arbre binaire en C).

1. Proposer une structure de données en *C* permettant d'implémenter un arbre binaire avec des étiquettes entières. Le nouveau type créé aura deux alias, `bintree` et `node`. Comment représenter l'arbre vide ?
2. Implémenter un constructeur `bintree *bintree_create(bintree *l, int v, bintree *r)` qui crée un arbre à partir de deux sous-arbres. Comment créer un arbre ne contenant qu'un seul nœud feuille ?
3. Pourquoi est-il important que `l` et `r` soient des arbres (donc des racines) et pas des pointeurs vers des nœuds quelconques ? Que se passe-t-il si `l` et `r` pointent vers le même arbre ? Dessinez !
4. Implémenter un destructeur `void bintree_free(bintree **t)` libérant tout l'espace mémoire alloué dans le tas pour stocker les données associées à l'arbre.
5. Dans le `main`, créer les objets correspondant aux arbres donnés en exemple ci-dessous.



Exercice 2 (Implémentation des fonctions de manipulation basiques).
Implémenter les fonctions de manipulation suivantes en C dans un fichier nommé `NOM_bintree.c`. Nous les avons codées la semaine dernière en OCaml.

```c
// constructeur
bintree *bintree_create(node *l, elt_type val, node *r);

//destructeur
void bintree_free(bintree **addr_t);

// accesseurs
int  bintree_number_of_nodes(bintree *t);
int  bintree_height(bintree *t);
int  bintree_leaves(bintree *t);
void bintree_print_preorder(bintree *t);
void bintree_print_inorder(bintree *t);
void bintree_print_postorder(bintree *t);
```

Les tests seront effectués en dur dans le `main`.

Exercice 3 (Liste préfixe/infixe/postfixe et compilation séparée).
En utilisant la bibliothèque de fonctions `mylist.h` et `mylist.c` fournie sur Moodle (code donné directement et non sous la forme d'une librairie dynamique), coder les trois fonctions supplémentaires suivantes.

```
list *bintree_preorder(bintree *t);
list *bintree_inorder(bintree *t);
list *bintree_postorder(bintree *t);
```

Vous ne copierez aucune fonction de la bibliothèque `mylist`. Vous procéderez en utilisant la compilation séparée.
Vous créerez notamment un script Shell `compile.sh` pour faciliter la compilation des fichiers source. Vous validerez les fonctions créées sur plusieurs tests, en comparant leurs résultats par rapport au précédentes fonction qui se contentaient d'afficher les différents parcours.