Partie 3 – Implementation

a) Codez les fonctions des arbres (et faites les tests) avec la structure interne suivante :
Ne pas oublier de créer fichiers séparés, c.-à-d., main.c, tree.c, tree.h.

```
struct NodeTree {
      int value;
      struct NodeTree* left;
      struct NodeTree* right;
}
```

Création :

1. `struct NodeTree* createEmptyTree();`
2. `struct NodeTree* bin(struct NodeTree* leftTree, int value, struct NodeTree* rightTree);`

Accès :

3. `bool isEmpty(struct NodeTree* tree);`
4. `int root(struct NodeTree* tree);`
5. `struct NodeTree* left(struct NodeTree* tree);`
6. `struct NodeTree* right(struct NodeTree* tree);`

Suppression :

7. `void deleteLeaf(struct NodeTree* tree); // efface une feuille`
8. `void deleteTree(struct NodeTree* tree); // efface tout l'arbre`

Parcours (Afficher les valeurs):

9. `void preorder(struct NodeTree* tree);`
10. `void postorder(struct NodeTree* tree);`
11. `void inorder(struct NodeTree* tree);`
12. `void BFS(struct NodeTree* tree); //(BFS=largeur)utiliser Queue!`

Autres operations :

13. `int numberNodes(struct NodeTree* tree);`
14. `int height(struct NodeTree* tree);`
15. `int numberLeaves(struct NodeTree* tree);`

```
16. int numberInternalNodes(struct NodeTree* tree);
17. int numberRepetitions(struct NodeTree* tree, int x);
18. void pathTo(struct NodeTree* tree, int x); // l'afficher
19. int sumValues(struct NodeTree* tree);
20. bool belongs(struct NodeTree* tree, int x);
21. int lastFullLevel(struct NodeTree* tree);
22. void descendants(struct NodeTree* tree, int x); // les afficher
23. struct NodeTree* mirror(struct NodeTree* tree);
24. bool equals(struct NodeTree* tree1, struct NodeTree* tree2);
```

Création d'un arbre à partir de parcours :

```
25. struct NodeTree* buildFromPreIn(int pre[],int in[],int sp,int
    ep,int si,int ei);
26. struct NodeTree* buildFromPostIn(int post[],int in[],int sp,int
    ep,int si,int ei)
27. struct NodeTree* buildFromBFSIn(int bfs[],int in[],int sb,int
    eb,int si,int ei);
```

→ Commentaires ←

- pre[],post[],in[],bfs[] sont des tableaux contenant les parcours preorder, postorder, inorder et largeur respectivement.
- tous les autres arguments sont pour vous aider à coder les fonctions d'une façon récursive. La notation «sx» et «ex» signifient indice début (s=start) et indice fin (e=end) du tableau «x», avec i=in, p=pre ou post et b=bfs.
- pour vous donner un example, si on a :

```
int pre[8] = {1,2,4,3,5,7,8,6};
int in[8] = {4,2,1,7,5,8,3,6};
```

on appellera la fonction dans le main comme :

```
struct NodeTree* tree = buildFromPreIn(pre,in,0,7,0,7);
```

puis les appels récursifs vont modifier les indices accordement.

- utilisez des fonctions auxiliaires pour calculer/mettre à jour les indices (vous allez avoir besoin)