BST – Implementation

In this lab, you will implement the binary search tree functions using dynamic allocation of nodes, the latters being defined as follows :

```
typedef struct _node {
  int val;
  struct _node *left, *right;
} node_t;
```

The list of functions to implement are :

```
void initTree(node_t **ptree) ; // initialize the tree's root node
int insertTree(node_t **ptree, int val) ; // add the node to tree
void inorderTree(node_t *ptree, int level) ; // inorder traversal of tree
void preorderTree(node_t *ptree, int level) ; // preorder traversal of tree
void postorderTree(node_t *ptree, int level); // postorder traversal of tree
void breadthTree(node_t *ptree) ; // breadth first traversal of tree

int maxTree(node_t *ptree) ; // find max value in tree
int minTree(node_t *ptree) ; // find min value in tree

node_t *searchTree(node_t *ptree, int val) ; // search for val in tree and return node

node_t *removeTree(node_t *ptree, int val) ; // remove val from tree and return node
```

Test the functions using similar code :

```
int main()
{
  node_t *myTree;

  initTree(&myTree);
  insertTree(&myTree, 50);
  insertTree(&myTree, 45);
  insertTree(&myTree, 65);
  insertTree(&myTree, 55);

  inorderTree(myTree, 0);
  preorderTree(myTree, 0);
  postorderTree(myTree, 0);


  printf("max = %d\n", maxTree(myTree));
  printf("min = %d\n", minTree(myTree));
  printf("nb of nodes = %d\n", nbNodesTree(myTree));
  printf("height tree = %d\n\n", heightTree(myTree));
  breadthTree(myTree);

  printf("search for 55 = %d\n", searchTree(myTree, 55)->val);
  node_t *pnd = searchTree(myTree, 77);
  printf("search for 77 = %p\n", pnd);


}
```

Bonus Exercise

Given a binary tree with an expression, find the algorithm for its evaluation, then implement it.
(Example given in class)