Binary Search Trees – Implementation

In this lab, you will be implementing 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;
```

BST Property : A node's value is bigger than its left child's value and smaller or equal to it's right child's value

The list of functions to implement are :

```
node_t *insertTree(node_t *ptree, int val) ; // add val to the tree & return the new tree
void inorderTree(node_t *ptree, int lvl) ; // inorder traversal of tree
void preorderTree(node_t *ptree, int lvl) ; //preorder traversal of tree
void postorderTree(node_t *ptree, int lvl); //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

int heightTree(node_t *ptree) ; // returns the height of the tree

int nbNodesTree(node_t *ptree) ; // returns the numbre of nodes in the tree

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

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

Test the functions using similar code :

```
int main() {
  node_t *myTree = NULL; // empty tree

  myTree = insertTree(myTree, 50);
  myTree = insertTree(myTree, 45);
  myTree = insertTree(myTree, 65);
  myTree = insertTree(myTree, 55);
  myTree = insertTree(myTree, 54);
  myTree = insertTree(myTree, 56);
  myTree = insertTree(myTree, 80);
  myTree = insertTree(myTree, 70);
  myTree = insertTree(myTree, 85);
  myTree = insertTree(myTree, 30);
  myTree = insertTree(myTree, 47) ;


  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);

  myTree = removeTree(myTree, 65) ;
}
```