DSA – Practical
Trees Operations

Continue what you worked on in previous tutorial to implement other trees operations:

1. Write the implementation of the following functions:

```
// Finds the node with value x in tree, and returns a pointer to it (NULL
if not found)
BinaryTreeNode* bstFind(BinaryTreeNode* root, double x) {
       …
}

// Returns a pointer to the node with minimum value in the tree (NULL if
tree empty)
BinaryTreeNode* bstMin(BinaryTreeNode* root) {
       …
}

// Returns a pointer to the node with maximum value in the tree (NULL if
tree empty)
BinaryTreeNode* bstMax(BinaryTreeNode* root) {
       …
}

// Clears all nodes from memory
void bstClear(BinaryTreeNode* root) {
       …
}
```

2. Test the above functions with something like the following main() function:

```
void main(){

        BinaryTreeNode* node;
        BinaryTreeNode* root = (BinaryTreeNode*) malloc(sizeof(BinaryTreeNode));
        root->key = 5.5;
        root->leftChild = NULL;
        root->rightChild = NULL;

        bstInsert(root, 7.7);
        bstInsert(root, 3.2);
        bstInsert(root, 4.0);
        ...

        node = bstFind(root, 3.2);
        if(node != NULL)
                printf("The value %f was found.\n", node->key);
        else
                printf("Value 3.2 was not found.\n";

        node = bstFind(root, 6.2);
        if(node != NULL)
                printf("The value %f was found.\n", node->key);
        else
                printf("Value 6.2 was not found.\n";

        node = bstMin(root);
        if(node != NULL)
                printf("The minimum value is %f.\n", node->key);

        node = bstMax(root);
        if(node != NULL)
                printf("The maximum value is %f.\n", node->key);


        printf("Height of binary search tree is %d.\n", bstHeight(root));

        bstClear(root);
        root = NULL;

}
```