15. **Binary Search Tree (Search, Min, Max)**

Aim: To implement a Binary Search Tree with search, minimum, and maximum operations.

Algorithm:

1. **Insert nodes in BST (left < root < right).**
2. **Search: Traverse left/right depending on key.**
3. **Min: Traverse left until NULL.**
4. **Max: Traverse right until NULL.**

code:

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int data;

   struct Node* left;

   struct Node* right;

};


struct Node* newNode(int data) {

   struct Node* node=(struct Node*)malloc(sizeof(struct Node));

   node->data=data; node->left=node->right=NULL;

   return node;

}


struct Node* insert(struct Node* root,int data) {
```

```
    if(root==NULL) return newNode(data);
    if(data<root->data) root->left=insert(root->left,data);
    else if(data>root->data) root->right=insert(root->right,data);
    return root;
}

struct Node* search(struct Node* root,int key) {
    if(root==NULL || root->data==key) return root;
    if(key<root->data) return search(root->left,key);
    return search(root->right,key);
}

int minValue(struct Node* root) {
    while(root->left) root=root->left;
    return root->data;
}

int maxValue(struct Node* root) {
    while(root->right) root=root->right;
    return root->data;
}

int main() {
    struct Node* root=NULL;
```

```
    root=insert(root,50);
    insert(root,30); insert(root,70);
    insert(root,20); insert(root,40); insert(root,60); insert(root,80);

    int key=40;
    if(search(root,key)) printf("Element %d found\n",key);
    else printf("Not found\n");
    printf("Min: %d\n",minValue(root));
    printf("Max: %d\n",maxValue(root));
    return 0;
}
```

Sample Output:

Element 40 found Min: 20

Max: 80

Result:

Search, min, and max operations were successfully implemented on a BST.