Assignment-4

1.Write a program to insert a node in a binary search tree.

Ans.Insert (TREE, ITEM)

Step 1: IF TREE = NULL. Allocate memory for TREE. SET TREE -> DATA = ITEM. SET TREE -> LEFT = TREE -> RIGHT = NULL. ELSE. IF ITEM < TREE -> DATA. Insert(TREE -> LEFT, ITEM) ELSE. Insert(TREE -> RIGHT, ITEM) [END OF IF] [END OF IF]

Step 2: END.

2.Write a program to search a node in the Binary Search Tree.

Ans./ C function to search a given key in a given BST

```
struct node* search(struct node* root, int key)
{

   // Base Cases: root is null or key is present at root

   if (root == NULL || root->key == key)

      return root;



   // Key is greater than root's key

   if (root->key < key)

      return search(root->right, key);


   // Key is smaller than root's key

   return search(root->left, key);
```

}

3.Write a program to Delete a node in the binary search tree.
Ans.Delete (TREE, ITEM)

```
Step 1: IF TREE = NULL
   Write "item not found in the tree" ELSE IF ITEM < TREE -> DATA
  Delete(TREE->LEFT, ITEM)
  ELSE IF ITEM > TREE -> DATA
   Delete(TREE -> RIGHT, ITEM)
  ELSE IF TREE -> LEFT AND TREE -> RIGHT
  SET TEMP = findLargestNode(TREE -> LEFT)
  SET TREE -> DATA = TEMP -> DATA
   Delete(TREE -> LEFT, TEMP -> DATA)
  ELSE
   SET TEMP = TREE
   IF TREE -> LEFT = NULL AND TREE -> RIGHT = NULL
   SET TREE = NULL
  ELSE IF TREE -> LEFT != NULL
  SET TREE = TREE -> LEFT
  ELSE
    SET TREE = TREE -> RIGHT
  [END OF IF]
  FREE TEMP
[END OF IF]
Step 2: END.
```

4.Write a program to find the minimum value in the Binary Search Tree.
Ans.

```
// C++ implementation of the approach
#include <bits/stdc++.h>

using namespace std;
```

```
/* A binary tree node has data, pointer to left child

   and a pointer to right child */

struct node {

    int data;

    struct node* left;

    struct node* right;
};

/* Helper function that allocates a new node

   with the given data and NULL left and right

   pointers. */

struct node* newNode(int data)
{

    struct node* node = (struct node*)

        malloc(sizeof(struct node));

    node->data = data;

    node->left = NULL;

    node->right = NULL;
```

```
    return (node);
}

/* Give a binary search tree and a number,
   inserts a new node with the given number in
   the correct place in the tree. Returns the new
   root pointer which the caller should then use
   (the standard trick to avoid using reference
   parameters). */
struct node* insert(struct node* node, int data)
{

    /* 1. If the tree is empty, return a new,
    single node */
    if (node == NULL)
        return (newNode(data));
    else {

        /* 2. Otherwise, recur down the tree */
```

```
        if (data <= node->data)
            node->left = insert(node->left, data);
        else
            node->right = insert(node->right, data);

        /* return the (unchanged) node pointer */
        return node;
    }
}

// Function to return the minimum node
// in the given binary search tree

int minValue(struct node* node)
{
    if (node->left == NULL)
        return node->data;
    return minValue(node->left);
}

// Driver code

int main()
{
```

```
    // Create the BST

    struct node* root = NULL;

    root = insert(root, 4);

    insert(root, 2);

    insert(root, 1);

    insert(root, 3);

    insert(root, 6);

    insert(root, 5);


    cout << minValue(root);


    return 0;
}
```

Output:

1