Lab program 8a:-

Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree.

```
#include <stdio.h>
#include <stdlib.h>

/* Structure of BST Node */
struct node
{
   int data;
   struct node *left;
   struct node *right;
};

struct node *root = NULL;

/* Function Prototypes */
struct node* insert(struct node *root, int data);
void inorder(struct node *root);
void preorder(struct node *root);
void postorder(struct node *root);

/* Main Function */
int main()
{
   int choice, value;

   while (1)
   {
      printf("\n\n----- BINARY SEARCH TREE MENU -----");
```

```
printf("\n1. Insert element");
printf("\n2. In-order Traversal");
printf("\n3. Pre-order Traversal");
printf("\n4. Post-order Traversal");
printf("\n5. Exit");
printf("\nEnter your choice: ");
scanf("%d", &choice);

switch (choice)
{
case 1:
   printf("Enter value to insert: ");
   scanf("%d", &value);
   root = insert(root, value);
   break;

case 2:
   printf("\nIn-order Traversal: ");
   inorder(root);
   break;

case 3:
   printf("\nPre-order Traversal: ");
   preorder(root);
   break;

case 4:
   printf("\nPost-order Traversal: ");
   postorder(root);
```

```
            break;

        case 5:
            exit(0);

        default:
            printf("\nInvalid choice!");
        }
    }
    return 0;
}

/* Insert a Node into BST */
struct node* insert(struct node *root, int data)
{
    if (root == NULL)
    {
        struct node *newnode;
        newnode = (struct node *)malloc(sizeof(struct node));
        newnode->data = data;
        newnode->left = newnode->right = NULL;
        return newnode;
    }

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

```
    return root;
}

/* In-order Traversal */
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

/* Pre-order Traversal */
void preorder(struct node *root)
{
    if (root != NULL)
    {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

/* Post-order Traversal */
void postorder(struct node *root)
{
    if (root != NULL)
```

```
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}
```

```
----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 1
Enter value to insert: 12


----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 1
Enter value to insert: 13


----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 1
Enter value to insert: 14


----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 2

In-order Traversal: 12 13 14
```

```
----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 3

Pre-order Traversal: 12 13 14

----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 4

Post-order Traversal: 14 13 12

----- BINARY SEARCH TREE MENU -----
1. Insert element
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 24.515 s
Press any key to continue.
```