```c
/*
  TITLE:IMPLEMENT BINARY SEARCH TREE
  NAME:Tauseef Mushtaque Ali Shaikh
  CLASS: S.Y.[CO]
  ROLLNO: 18CO63
  SUBJECT: DS
  DATE: 30/9/19
  DISCRIPTION: In this Program binary search is implemented i.e. inorder,
  preorder, postorder.
*/
#include<stdio.h>
#include<stdlib.h>

struct bstree
{
int data;
struct bstree *left,*right;
};

struct bstree *insert(struct bstree *root, int d)
{
struct bstree *p;
if(root==NULL)
{
p=(struct bstree *)malloc(sizeof(struct bstree));
p->data=d;
p->left=NULL;
p->right=NULL;
root=p;
}
else
{
if(d<root->data)
root->left=insert(root->left,d);
else
root->right=insert(root->right,d);
}
return root;
}

void preorder(struct bstree *r)
{
    if(r!=NULL)
    {
        printf("\n%d",r->data);
        preorder(r->left);
        preorder(r->right);
        }
}

void inorder(struct bstree *r)
{
    if(r!=NULL)
    {
        inorder(r->left);
        printf("\n%d",r->data);
        inorder(r->right);
        }
}

```

```
60  void postorder(struct bstree *r)
61  {
62      if(r!=NULL)
63      {
64          postorder(r->left);
65          postorder(r->right);
66          printf("\n%d",r->data);
67          }
68  }
69
70  struct bstree *search(struct bstree *root, int key)
71  {
72      if(root!=NULL)
73      {
74          if(key==root->data)
75          return root;
76          else
77          {
78              if(key<root->data)
79              root->left=search(root->left,key);
80              else
81              root->right=search(root->right,key);
82              }
83          }
84      else
85      return NULL;
86  }
87
88  int main()
89  {
90      int ch,d;
91      struct bstree *p, *root=NULL;
92      while(1)
93      {
94          printf("\n\n\t\t\tMENU\n1. INSERT\n2. INORDER\n3. PREORDER\n4.
        POSTORDER\n5. SEARCH\n0. EXIT\n");
95          printf("ENTER YOUT CHOICE: ");
96          scanf("%d", &ch);
97          switch(ch)
98          {
99              case 1:
100             printf("\nENTER DATA: ");
101             scanf("%d",&d);
102             root=insert(root,d);
103             break;
104
105             case 2:
106             printf("\n\tINORDER TRANSVERSAL IS: ");
107             inorder(root);
108             break;
109
110             case 3:
111             printf("\n\tPREORDER TRANSVERSAL IS: ");
112             preorder(root);
113             break;
114
115             case 4:
116             printf("\n\tPOSTORDER TRANSVERSAL IS: ");
117             postorder(root);
118             break;
```

```
            case 5:
            printf("ENTER KEY: ");
            scanf("%d", &d);
            p=search(root,d);
            if(p==NULL)
            printf("\nGIVEN KEY DOES NOT EXIST!");
            else
            printf("\nGIVEN KEY DOES EXIST!");

            case 0:
            exit(0);
            break;

            default:
            printf("INVALID CHOICE");
            }
        }
    }
```