EX. NO: 6 **IMPLEMENTATION OF BINARY SEARCH TREE**

AIM:

To write a C program to implementation of binary search tree.

DESCRIPTION:

A Binary Search Tree (BST) is a tree in which all the nodes follow the below- mentioned properties

The left sub-tree of a node has a key less than or equal to its parent node's key.

The right sub-tree of a node has a key greater than to its parent node's key.

Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as left_subtree (keys) ≤ node (key) ≤ right_subtree (keys)

Representation

BST is a collection of nodes arranged in a way where they maintain BST properties. Each node has a key and an associated value. While searching, the desired key is compared to the keys in BST and if found, the associated value is retrieved.

Following is a pictorial representation of BST

Basic Operations

Following are the basic operations of a tree

- Search − Searches an element in a tree.
- Insert − Inserts an element in a tree.
- Pre-order Traversal − Traverses a tree in a pre-order manner.
- In-order Traversal − Traverses a tree in an in-order manner.
- Post-order Traversal − Traverses a tree in a post-order manner.

ALGORITHM:

1. Declare function create (), search (), delete (), Display ().
2. Create a structure for a tree contains left pointer and right pointer.

3. Insert an element is by checking the top node and the leaf node and the operation will be performed.
4. Deleting an element contains searching the tree and deleting the item.
5. Display the Tree elements.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<alloc.h>

struct
tree
{ int data; struct
      tree *lchild;
      struct tree
      *rchild;
}*t,*temp;
 int
element
;
void inorder(struct tree *); void
preorder(struct tree *); void
postorder(struct tree *); struct
tree * create(struct tree *, int);
struct tree * find(struct tree *,
int); struct tree * insert(struct
tree *, int); struct tree *
del(struct tree *, int); struct tree
* findmin(struct tree *); struct
tree * findmax(struct tree *); void
main()
{ int
      ch
      ;
 do
```

```
{ printf("\n\t\t\tBINARY SEARCH
      TREE"); printf("\n\t\t\t******
      ****** ****"); printf("\nMain
      Menu\n");
  printf("\n1.Create\n2.Insert\n3.Delete\n4.Find\n5.FindMin\n6.FindMax")
;
      printf("\n7.Inorder\n8.Preorder\n9.Postorder\n10.Exit
      \n"); printf("\nEnter ur choice :"); scanf("%d",&ch);
      switch(ch)
      { case
            1:
                  printf("\nEnter the
                  data:");
                  scanf("%d",&element);
                  t=create(t,element);
                  inorder(t);

                  br
                  ea
                  k;
            case 2:
                  printf("\nEnter the
                  data:");
                  scanf("%d",&element);
                  t=insert(t,element);
                  inorder(t); break;
            case 3:
                  printf("\nEnter the
                  data:");
                  scanf("%d",&element);
                  t=del(t,element);
                  inorder(t); break;
            case 4:
                  printf("\nEnter the data:");
                  scanf("%d",&element); temp=find(t,element);
                  if(temp->data==element) printf("\nElement %d
                  is at %d",element,temp);
                  else printf("\nElement is not
                        found");
```

```
                        break;
                  case 5:
                        temp=findmin(t);
                        printf("\nMax element=%d",temp->data);
                        break;
                  case 6:
                        temp=findmax(t);
                        printf("\nMax element=%d",temp->data);
                        break;
                  case 7:
                        inorder(
                        t);
                        break;
                  case 8:
                        preorder(t
                  ); break;
                  case 9:
                  postorder(t);
                  break;
                  case 10:
                        exit(0);
            }
      }while(ch<=10);
}
struct tree * create(struct tree *t, int element)
{ t=(struct tree *)malloc(sizeof(struct
tree)); t->data=element; t->lchild=NULL; t-
>rchild=NULL; return t; }
struct tree * find(struct tree *t, int element)
{ if(t==NULL)
      return NULL;
      if(element<t->data) return(find(t-
            >lchild,element));
      else if(element>t->data) return(find(t-
            >rchild,element));
            else
                  retu
                  rn
                  t;
```

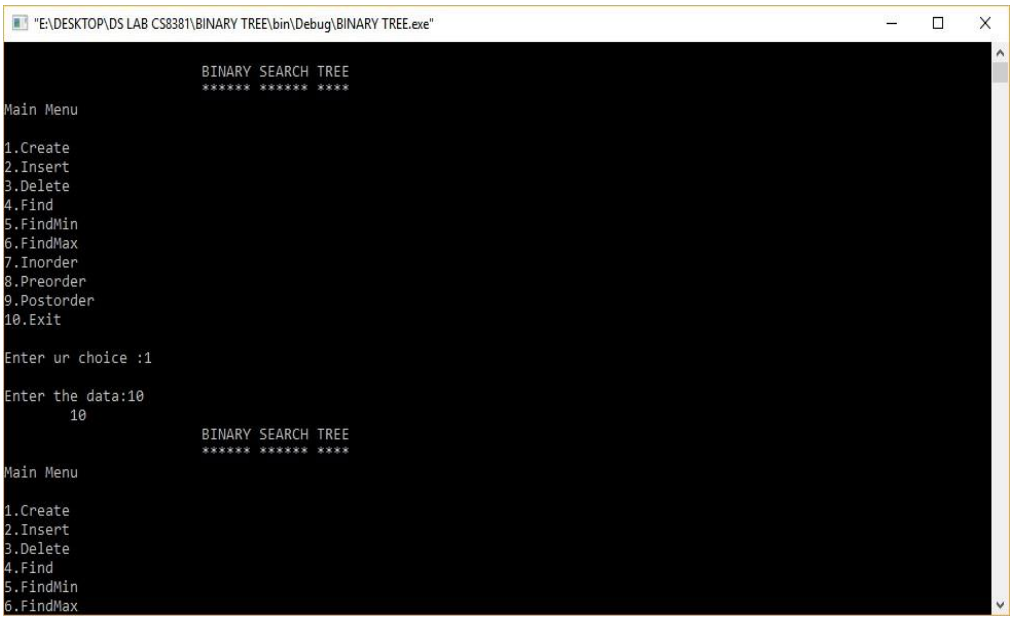
```
}
struct tree *findmin(struct tree *t)
{ if(t==NULL)
      return NULL;
      else if(t-
            >lchild==NULL)
            return t;
            else return(findmin(t-
                  >lchild));
}
struct tree *findmax(struct tree *t)
{
      if(t!=
      NULL)


      { while(t-
            >rchild!=NULL)
            t=t->rchild;
      }
return t;
}
struct tree *insert(struct tree *t,int element)
{
      if(t==
      NULL)
      {

            t=(struct tree *)malloc(sizeof(struct
            tree)); t->data=element; t->lchild=NULL;
            t->rchild=NULL; return t;
      }
      e
      l
      s
      e
      { if(element<t-
            >data)
```

```
            { t->lchild=insert(t-
                  >lchild,element);
            } else if(element>t-
            >data)
                  { t->rchild=insert(t-
                        >rchild,element);
                  } else
                  if(element==t-
                  >data)
                  { printf("element already
                        present\n");
                  }
                  retu
                  rn
                  t;
      }
}
struct tree * del(struct tree *t, int element)
{ if(t==NULL) printf("element not
      found\n");
       else if(element<t->data) t-
      >lchild=del(t->lchild,element);
            else if(element>t->data) t-
                  >rchild=del(t->rchild,element);
                  else if(t->lchild&&t-
                        >rchild)
                        { temp=findmin(t->rchild); t-
                              >data=temp->data; t-
                              >rchild=del(t->rchild,t-
                              >data);
                        }
                        e
                        l
                        s
                        e
                        {
                        t
                        e
                        m
```

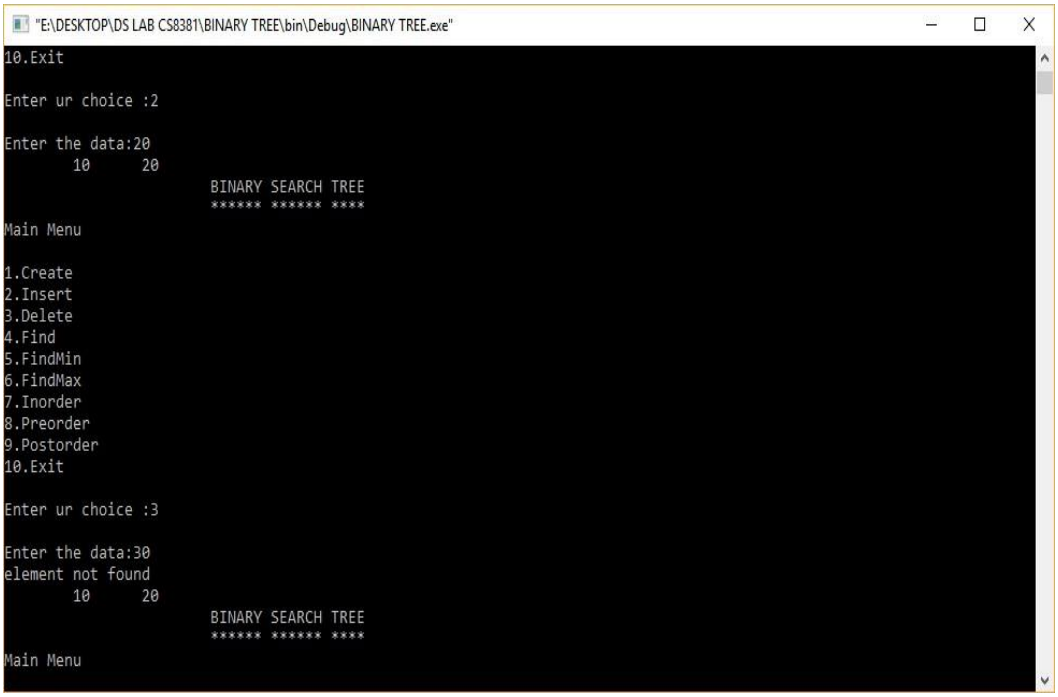
```
p
=
t
;
i
f
(
t
-
>
l
c
h
i
l
d
=
=
N
U
L
L
)
t
=
t
-
>
r
c
h
i
l
d
;
      else if(t-
            >rchild==NULL)
            t=t->lchild;
```

```
                                        free(tem
                        p); }
        return t;
}
void inorder(struct tree *t)
{ if(t==NULL)
        return;
        else
        { inorder(t->lchild);
                printf("\t%d",t-
                >data); inorder(t-
                >rchild);
        }
}
void preorder(struct tree *t)
{ if(t==NULL)
        return;
        else
        { printf("\t%d",t-
                >data);
                preorder(t-
                >lchild);
                preorder(t-
                >rchild);
        }
}
void postorder(struct tree *t)
{ if(t==NULL)
        return;
        else
        { postorder(t->lchild);
                postorder(t-
                >rchild);
                printf("\t%d",t-
                >data);}}
```

OUTPUT:


"E:\DESKTOP\DS LAB CS8381\BINARY TREE\bin\Debug\BINARY TREE.exe"
BINARY SEARCH TREE
****** ****** ****
Main Menu
1.Create
2.Insert
3.Delete
4.Find
5.FindMin
6.FindMax
7.Inorder
8.Preorder
9.Postorder
10.Exit
Enter ur choice :1
Enter the data:10
10
BINARY SEARCH TREE
****** ****** ****
Main Menu
1.Create
2.Insert
3.Delete
4.Find
5.FindMin
6.FindMax


"E:\DESKTOP\DS LAB CS8381\BINARY TREE\bin\Debug\BINARY TREE.exe"
Enter the data:25
10 20 25
BINARY SEARCH TREE
****** ****** ****
Main Menu
1.Create
2.Insert
3.Delete
4.Find
5.FindMin
6.FindMax
7.Inorder
8.Preorder
9.Postorder
10.Exit
Enter ur choice :2
Enter the data:30
10 20 25 30
BINARY SEARCH TREE
****** ****** ****
Main Menu
1.Create
2.Insert
3.Delete
4.Find


"E:\DESKTOP\DS LAB CS8381\BINARY TREE\bin\Debug\BINARY TREE.exe"
10.Exit
Enter ur choice :2
Enter the data:20
10 20
BINARY SEARCH TREE
****** ****** ****
Main Menu
1.Create
2.Insert
3.Delete
4.Find
5.FindMin
6.FindMax
7.Inorder
8.Preorder
9.Postorder
10.Exit
Enter ur choice :3
Enter the data:30
element not found
10 20
BINARY SEARCH TREE
****** ****** ****
Main Menu

RESULT:

Thus the C program for binary search tree was created, executed and output was verified successfully.