Ex.No: 9 Binary Search Tree(BST)- Implementation

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct bst
{
 int data;
 struct bst *left, *right;
}node;
void insert(node *,node *);
node *search(node *,int,node **);
int findmax(node *);
int findmin(node *);
void del(node *,int);
void display(node *,int);
void main()
{
 int ch;
 char ans='n';
 int key,a,b;
 node *new,*root,*temp,*parent;
 node *get_node();
 clrscr();
 root=NULL;
 do
 {
  printf("\n 1.Insert \n 2.Search \n 3.Findmin \n 4.Findmax \n 5.Delete \n 6.Display \n 7.Exit ");
  printf("\n Enter your choice:");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1:
          do
          {
           new=get_node();
           printf("\n Enter the element:");
           scanf("%d",&new->data);
           if(root==NULL)
            root=new;
           else
            insert(root,new);
           printf("Do you wants to continue?(Y/N):");
           ans=getch();
          }while(ans=='y');
          break;
   case 2:
          printf("\n Enter the element which you want to search:");
          scanf("%d",&key);
          temp=search(root,key,&parent);
          if(temp==root)
           printf("\n No parent node for root");
          else
           printf("\n Parent of %d is %d",temp->data,parent->data);
          break;
  case 3:
          a=findmin(root);
          printf("\n The smallest element found in the tree is %d",a);
          break;
  case 4:
          b=findmax(root);
          printf("\n The biggest element found in the treee is %d",b);
          break;
  case 5:
          printf("\n Enter the element which you want to delete:");
          scanf("%d",&key);
          del(root,key);
          break;
  case 6:
          display(root,1);
          break;
  case 7:
          exit(0);
          break;
  }
 }while(ch!=7);
}
node*get_node()
{
 node *temp;
 temp=(node*)malloc(sizeof(node));
 temp->left=NULL;
 temp->right=NULL;
 return temp;
}
void insert(node *root,node*new)
{
 if(new->data<root->data)
 {
  if(root->left==NULL)
   root->left=new;
  else
   insert(root->left,new);
 }
 if(new->data>root->data)
 {
  if(root->right==NULL)
   root->right=new;
  else
```

```
   insert(root->right,new);
  }
 }
node*search(node *root,int key,node **parent)
{
 node *temp;
 temp=root;
 while(temp!=NULL)
  {
  if(temp->data==key)
   {
    printf("%d is present",temp->data);
    return temp;
   }
  *parent=temp;
  if(temp->data>key)
   temp=temp->left;
  else
   temp=temp->right;
 }
 return NULL;
}
int findmin(node *tree)
{
 if(tree!=NULL)
  {
  while(tree->left!=NULL)
   {
    tree=tree->left;
   }
  }
 return tree->data;
}
int findmax(node *tree)
{
 if(tree==NULL)
  return NULL;
 else if(tree->right==NULL)
  return tree->data;
 else
  return findmax(tree->right);
}
void del(node*root,int key)
{
 node *temp,*parent,*suc,*suc1;
 temp=search(root,key,&parent);
 if(temp==NULL)
  {
  printf("\n Element not found");
  return;
  }
```

```
else
{
 if(temp->left!=NULL && temp->right!=NULL)
 {
  parent=temp;
  suc=temp->right;
  if(suc->left!=NULL)
  {
   while(suc->left!=NULL)
   {
    parent=suc;
    suc=suc->left;
   }
   temp->data=suc->data;
   parent->left=NULL;
  }
  else
  {
   temp->right=suc->right;
   temp->data=suc->data;
   free(suc);
  }
  printf("\n Now deleted it.");
  return;
 }
 if(temp->left==NULL && temp->right!=NULL)
 {
  if(parent->left==temp)
   parent->left=temp->right;
  else
   parent->right=temp->right;
  temp=NULL;
  free(temp);
  printf("\n Now deleted it.");
  return;
 }
 if(temp->left!=NULL && temp->right==NULL)
 {
  if(parent->left==temp)
   parent->left=temp->left;
  else
   parent->right=temp->left;
  temp=NULL;
  free(temp);
  printf("\n Now deleted it.");
  return;
 }
 if(temp->left==NULL && temp->right==NULL)
 {
  if(parent->left==temp)
   parent->left=NULL;
   else
    parent->right=NULL;
   printf("\n Now deleted it.");
   return;
  }
 }
}
void display(node*T,int level)
{
 int i;
 if(T!=NULL)
 {
  display(T->right,level+1);
  printf("\n");
  for(i=0;i<level;i++)
   printf("  ");
  printf("%d",T->data);
  display(T->left,level+1);
 }
}
```

OUTPUT

```
1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 1

ENTER THE ELEMENT: 4
DO U WANT TO CONTINUE?(y/n):
ENTER THE ELEMENT: 6
DO U WANT TO CONTINUE?(y/n):
ENTER THE ELEMENT: 2
DO U WANT TO CONTINUE?(y/n):
ENTER THE ELEMENT: 7
DO U WANT TO CONTINUE?(y/n):
ENTER THE ELEMENT: 1
DO U WANT TO CONTINUE?(y/n):
ENTER THE ELEMENT: 5
DO U WANT TO CONTINUE?(y/n):
ENTER THE ELEMENT: 3
DO U WANT TO CONTINUE?(y/n):


1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 6
```

```
    7
  6
    5
 4
    3
  2
    1
1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 2

ENTRE THE ELEMENT WHICH U WANT TO SEARCH: 6
6 IS PRESENT
PARENT OF 6 IS 4.

1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 2

ENTRE THE ELEMENT WHICH U WANT TO SEARCH: 4
4 IS PRESENT
No Parent node for Root.

1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 3

The smallest element found in the tree is 1.

1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 4

The Biggest element found in the tree is 7.

1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
```

```
ENTER YOUR CHOICE: 5

ENTER THE ELEMENT WHICH U WANT TO DELETE: 6
6 IS PRESENT
NOW DELETED IT...

1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 5

ENTER THE ELEMENT WHICH U WANT TO DELETE: 8
Element Not Found.

1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 6

    7
      5
  4
      3
    2
      1
1.INSERT
2.SEARCH
3.FINDMIN
4.FINDMAX
5.DELETE
6.DISPLAY
7.EXIT
ENTER YOUR CHOICE: 7
```