Lab 10

LAB 10:-

Program:-

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.

Program:-

```
#include<stdio.h>

#include<stdio.h>

#include<stdlib.h>

struct node

  {

    int info;

    struct node *rlink;

    struct node *llink;

  };

typedef struct node *NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

  {

    printf("mem full\n");

    exit(0);

  }

  return x;

}

void freenode(NODE x)
```

```
{
free(x);
}
NODE insert(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
  return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
  prev->llink=temp;
else
  prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
```

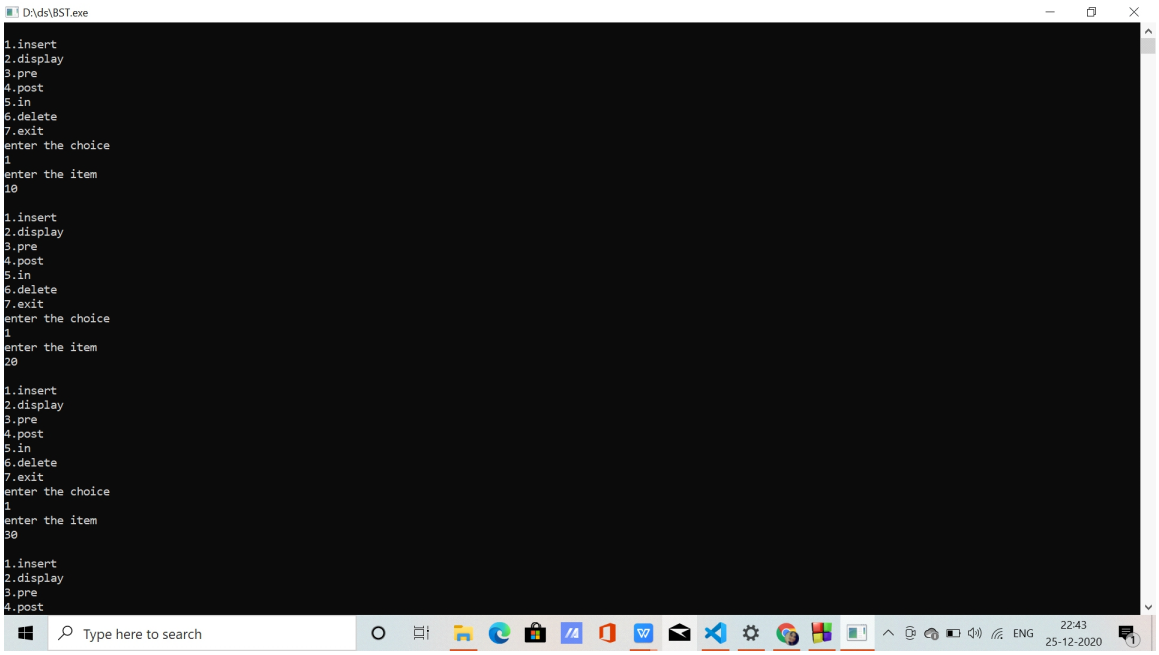
```
if(root!=NULL)
  {
    display(root->rlink,i+1);
    for(j=0;j<i;j++)
                printf("    ");
      printf("%d\n",root->info);
              display(root->llink,i+1);
  }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
  printf("not found\n");
  return root;
```

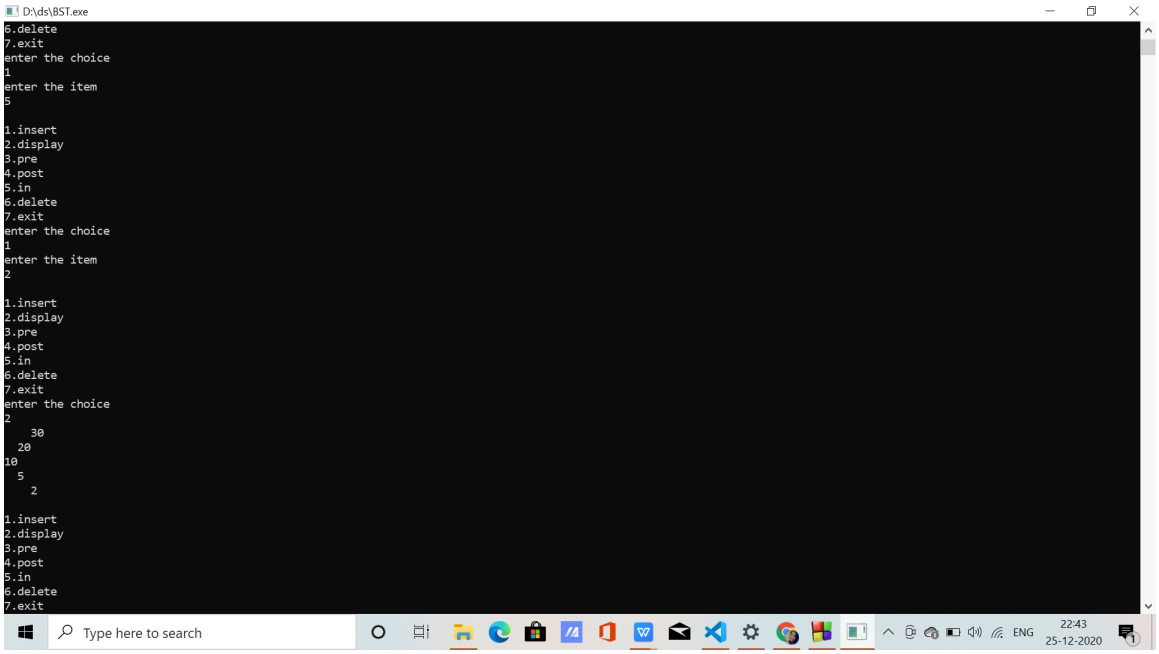
```
}
if(cur->llink==NULL)
  q=cur->rlink;
else if(cur->rlink==NULL)
  q=cur->llink;
else
  {
  suc=cur->rlink;
  while(suc->llink!=NULL)
    suc=suc->llink;
  suc->llink=cur->llink;
  q=cur->rlink;
  }
  if(parent==NULL)
    return q;
  if(cur==parent->llink)
    parent->llink=q;
  else
    parent->rlink=q;
  freenode(cur);
  return root;
  }

void preorder(NODE root)
{
if(root!=NULL)
  {
    printf("%d\n",root->info);
```

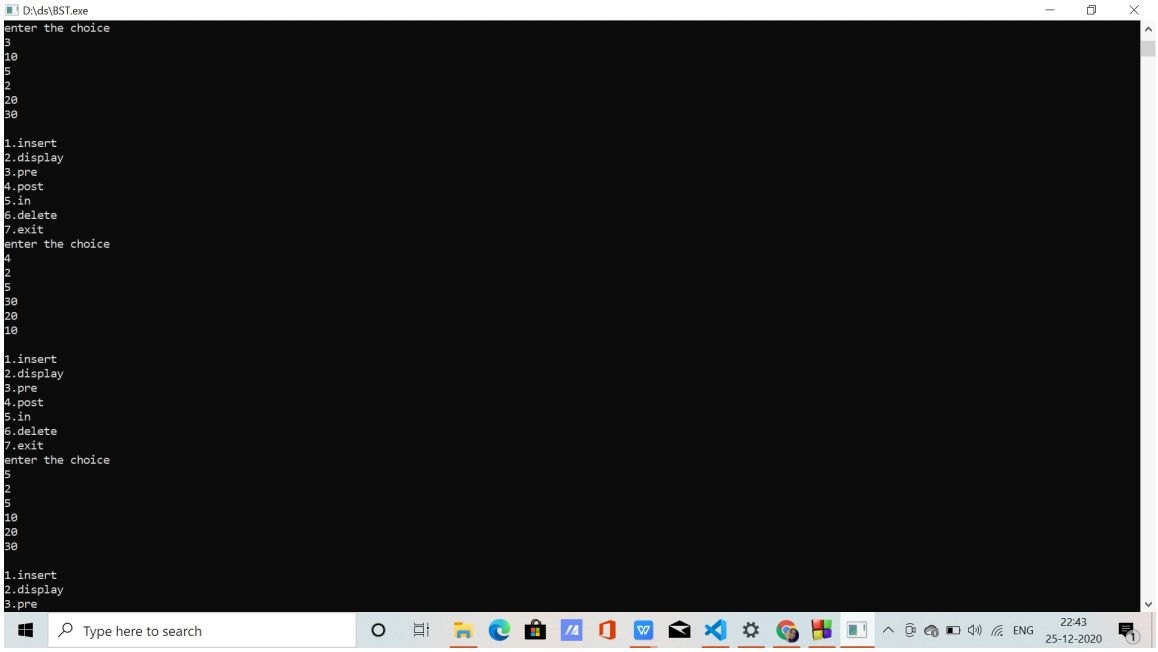
```
    preorder(root->llink);
    preorder(root->rlink);
    }
  }
void postorder(NODE root)
{
if(root!=NULL)
  {

    postorder(root->llink);
    postorder(root->rlink);
    printf("%d\n",root->info);
    }
  }
void inorder(NODE root)
{
if(root!=NULL)
  {

    inorder(root->llink);
    printf("%d\n",root->info);
    inorder(root->rlink);
    }
  }
int main()
{
int item,choice;
NODE root=NULL;
```

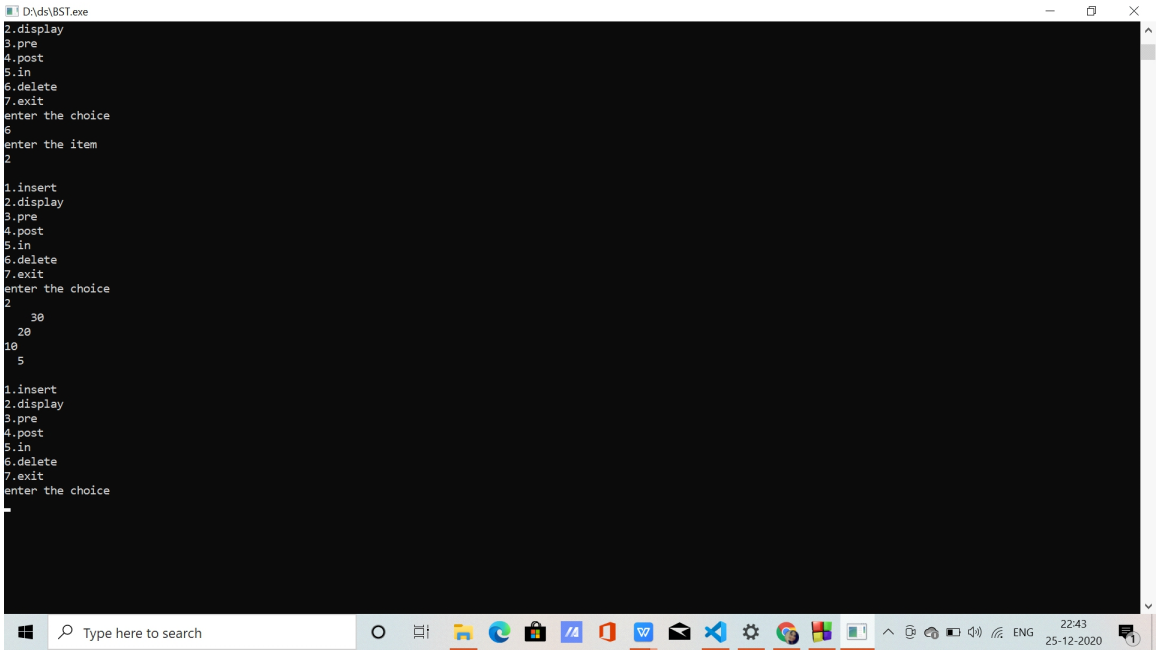
```
for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
   case 1:printf("enter the item\n");
                        scanf("%d",&item);
                        root=insert(root,item);
                        break;
   case 2:display(root,0);
                        break;
   case 3:preorder(root);
                        break;
   case 4:postorder(root);
                        break;
   case 5:inorder(root);
                        break;
   case 6:printf("enter the item\n");
                        scanf("%d",&item);
                        root=delete(root,item);
                        break;
   default:exit(0);
                          break;
             }
          }
 }
```

Screenshots:-






D:\ds\BST.exe
enter the choice
3
10
5
2
20
30
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
2
5
30
20
10
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
2
5
10
20
30
1.insert
2.display
3.pre
Type here to search
ENG
22:43
25-12-2020


D:\ds\BST.exe
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
2
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
30
20
10
5
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
Type here to search
ENG
22:43
25-12-2020

Written:-

Lab 10

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int info;
struct node *rlink;
struct node * llink;
};
typedef struct node * NODE;

NODE getnode()
{ NODE x;
x = (NODE) malloc (sizeof(struct node));
if (x==NULL)
{
pf("memfull\n");
}
void freenode(NODE x)
{
free(x);
}
NODE insert(NODE root, int item)
{
NODE temp, cur, prev;
temp=getnode();
temp ->rlink=NULL;
temp -> llink=NULL;
temp->info=item;
if (root==NULL)
return temp;
```

```
prev = NULL;
cur = root;
while (cur != NULL)
{
prev = cur;
~~cur = cur -> link;~~
cur = (item < cur -> info)? cur -> llink : cur -> rlink;
}
if (~~temp~~ item < prev -> info)
prev -> llink = temp;
else
prev -> rlink = temp;
return root;
}
void display (NODE root, int i)
{
int j;
if (root != NULL)
{
display (root -> rlink, i+1);
for (j=0; j<i; j++)
printf ("  ");
printf ("%d\n", root -> info);
display (root -> llink, i+1);
}
}
NODE delete (NODE root, int item)
{
NODE cur, parent, q, suc;
if (root == NULL)
{
printf ("empty tree");
```

```
return root;
}
parent = NULL;
cur = root;
while (cur != NULL && item != cur->info)
{
parent = cur
cur = (item < cur->info) ? cur->llink : cur->rlink;
}
if (cur == NULL)
pf ("not found \n");
return root;
}
if (cur->llink == NULL)
q = cur->rlink;
else if (cur->rlink == NULL)
q = cur->llink;
else
{
suc = cur->rlink;
while (suc->llink != NULL)
suc = suc->llink;
suc->llink = cur->llink;
q = cur->rlink;
}
freenode(cur);
return root;
}
```

```
}
Pf("\n1. insert \n2 display \n3 pre order \n4 post
\n5 in \n6 delete \n7 exit \n");
Pf ("enter the choice \n");
Sf("%d", &choice);
switch (choice)
{
case 1: printf (" enter the item \n");
    sf ("%d", &item)
    root = insert (root, item);
    break;
case 2: display (root, 0)
    break;
case 3: preorder (root);
    break;
case 4: postorder (root);
    break;
case 5: inorder (root);
    break;
case 6: printf (" enter the item \n");
    sf ("%d", &item);
    root = delete (root, item)
    break;
default: exit(0);
    break;
  }
 }
}
```