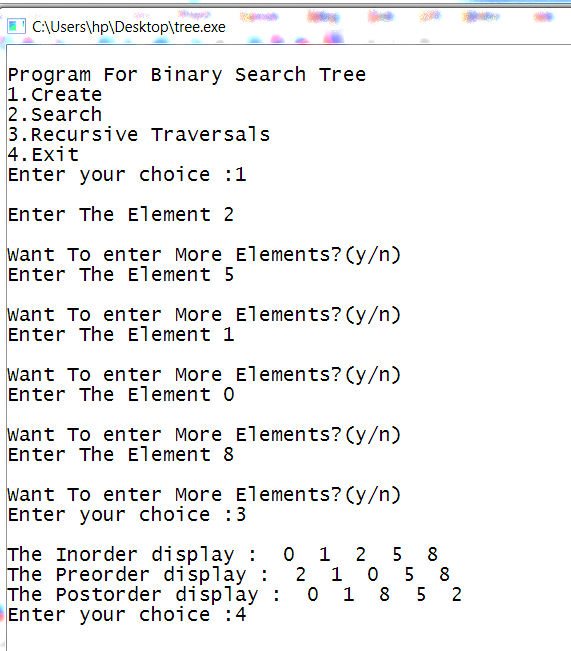
OUTPUT :



**PROGRAM TO IMPLEMENT BINARY SEARCH TREE AND TREE TRAVERSALS**

CODING:

# include <stdio.h>

# include <conio.h>

# include <stdlib.h>

typedef struct BST

{

int data;

struct BST \*lchild,\*rchild;

}node;

void insert(node \*,node \*);

void inorder(node \*);

void preorder(node \*);

void postorder(node \*);

node \*search(node \*,int,node \*\*);

int main()

{

int choice;

char ans='N';

int key;

node \*new\_node,\*root,\*tmp,\*parent;

node \*get\_node();

root=NULL;

printf("\nProgram For Binary Search Tree ");

printf("\n1.Create");

printf("\n2.Search");

printf("\n3.Recursive Traversals");

printf("\n4.Exit");

do

{

printf("\nEnter your choice :");

scanf("%d",&choice);

switch(choice)

{

case 1:

do

{

new\_node=get\_node();

printf("\nEnter The Element ");

scanf("%d",&new\_node->data);

if(root==NULL) /\* Tree is not Created \*/

root=new\_node;

else

insert(root,new\_node);

printf("\nWant To enter More Elements?(y/n)");

ans=getch();

}while(ans=='y');

break;

case 2:

printf("\nEnter Element to be searched :");

scanf("%d",&key);

tmp = search(root,key,&parent);

printf("\nParent of node %d is %d",

tmp->data,parent->data);

break;

case 3:

if(root==NULL)

printf("Tree Is Not Created");

else

{

printf("\nThe Inorder display : ");

inorder(root);

printf("\nThe Preorder display : ");

preorder(root);

printf("\nThe Postorder display : ");

postorder(root);

}

break;

}

}while(choice!=4);

getch();

return 0;

}

/\*

Get new Node

\*/

node \*get\_node()

{

node \*temp;

temp=(node \*)malloc(sizeof(node));

temp->lchild=NULL;

temp->rchild=NULL;

return temp;

}

/\*

This function is for creating a binary search tree

\*/

void insert(node \*root,node \*new\_node)

{

if(new\_node->data < root->data)

{

if(root->lchild==NULL)

root->lchild = new\_node;

else

insert(root->lchild,new\_node);

}

if(new\_node->data > root->data)

{

if(root->rchild==NULL)

root->rchild=new\_node;

else

insert(root->rchild,new\_node);

}

}

/\*

This function is for searching the node from

binary Search Tree

\*/

node \*search(node \*root,int key,node \*\*parent)

{

node \*temp;

temp=root;

while(temp!=NULL)

{

if(temp->data==key)

{

printf("\n The %d Element is Present",temp->data);

return temp;

}

\*parent=temp;

if(temp->data>key)

temp=temp->lchild;

else

temp=temp->rchild;

}

return NULL;

}

/\*

This function displays the tree in inorder fashion

\*/

void inorder(node \*temp)

{

if(temp!=NULL)

{

inorder(temp->lchild);

printf(" %d ",temp->data);

inorder(temp->rchild);

}

}

/\*

This function displays the tree in preorder fashion

\*/

void preorder(node \*temp)

{

if(temp!=NULL)

{

printf(" %d ",temp->data);

preorder(temp->lchild);

preorder(temp->rchild);

}

}

/\*

This function displays the tree in postorder fashion

\*/

void postorder(node \*temp)

{

if(temp!=NULL)

{

postorder(temp->lchild);

postorder(temp->rchild);

printf(" %d ",temp->data);

}

}