**NAME: CHIRAG SATAPATHY**

**REG. No.: 19BEI0107**

**COURSE CODE: CSE2003**

**SLOT: L27 + L28**

**DATA STRUCTURES AND ALGORITHMS**

**DIGITAL ASSIGNMENT – 5**

**1. Menu driven C program to create binary search tree. Perform insertion and deletion operations. Display the contents of BST using preorder, inorder and postorder traversal.**

**PSEUDOCODE:**

struct tree

{

int info

struct tree \*left

struct tree \*right

};

struct tree \*insert(struct tree \*,int)

void inorder(struct tree \*)

void postorder(struct tree \*)

void preorder(struct tree \*)

struct tree \*delet(struct tree \*,int)

int main()

{

struct tree \*root

int choice, item,item\_no

root = NULL

do

{

do

{

printf("\n\t 1. Insert in Binary Tree ")

printf("\n\t 2. Delete from Binary Tree ")

printf("\n\t 3. Inorder traversal of Binary tree")

printf("\n\t 4. Postorder traversal of Binary tree")

printf("\n\t 5. Preorder traversal of Binary tree")

printf("\n\t 6. Exit ")

printf("\n\t Enter choice : ")

scanf(" %d",&choice)

if(choice<1 || choice>7)

printf("\n Invalid choice - try again")

}while (choice<1 || choice>6)

switch(choice)

{

case 1:

printf("\n Enter new element: ")

scanf("%d", &item)

root= insert(root,item)

printf("\n root is %d",root->info)

printf("\n Inorder traversal of binary tree is : ")

inorder(root)

printf("\n Preorder traversal of binary tree is : ")

preorder(root)

printf("\n Postorder traversal of binary tree is : ")

postorder(root)

printf("\n")

break

case 2:

printf("\n Enter the element to be deleted : ")

scanf(" %d",&item\_no)

root=delet(root,item\_no)

printf("\n Inorder traversal of binary tree is : ")

inorder(root)

printf("\n Preorder traversal of binary tree is : ")

preorder(root)

printf("\n Postorder traversal of binary tree is : ")

postorder(root)

printf("\n")

break

case 3:

printf("\n Inorder traversal of binary tree is : ")

inorder(root)

break

case 4:

printf("\n Postorder traversal of binary tree is : ")

postorder(root)

break

case 5:

printf("\n Preorder traversal of binary tree is : ")

preorder(root)

break

default:

printf("\n End of program ")

}

}

while(choice !=6)

return(0)

}

struct tree \*insert(struct tree \*root, int x)

{

if(!root)

{

root=(struct tree\*)malloc(sizeof(struct tree))

root->info = x

root->left = NULL

root->right = NULL

return(root)

}

if(root->info > x)

root->left = insert(root->left,x)

else

{

if(root->info < x)

root->right = insert(root->right,x)

}

return(root)

}

void inorder(struct tree \*root)

{

if(root != NULL)

{

inorder(root->left)

printf(" %d",root->info)

inorder(root->right)

}

return;

}

void postorder(struct tree \*root)

{

if(root != NULL)

{

postorder(root->left)

postorder(root->right)

printf(" %d",root->info)

}

return

}

void preorder(struct tree \*root

{

if(root != NULL)

{

printf(" %d",root->info)

preorder(root->left)

preorder(root->right)

}

return

}

struct tree \*delet(struct tree \*ptr,int x)

{

struct tree \*p1,\*p2

if(!ptr)

{

printf("\n Node not found ")

return(ptr)

}

else

{

if(ptr->info < x)

{

ptr->right = delet(ptr->right,x);

/\*return(ptr);\*/

}

else if (ptr->info >x)

{

ptr->left=delet(ptr->left,x)

return ptr

}

else

{

if(ptr->info == x)

{

if(ptr->left == ptr->right)

{

free(ptr)

return(NULL)

}

else if(ptr->left==NULL)

{

p1=ptr->right

free(ptr)

return p1

}

else if(ptr->right==NULL)

{

p1=ptr->left

free(ptr)

return p1

}

else

{

p1=ptr->right

p2=ptr->right

while(p1->left != NULL)

p1=p1->left

p1->left=ptr->left

free(ptr)

return p2

}

}

}

}

return(ptr)

}

**CODE:**

#include<stdio.h>

#include<conio.h>

struct tree {

int info;

struct tree \*left;

struct tree \*right;

};

struct tree \*insert(struct tree \*,int);

void inorder(struct tree \*);

void postorder(struct tree \*);

void preorder(struct tree \*);

struct tree \*delet(struct tree \*,int);

int main()

{

struct tree \*root;

int choice, item,item\_no;

root = NULL;

/\* rear = NULL;\*/

do

{

do

{

printf("\n\t 1. Insert in Binary Tree ");

printf("\n\t 2. Delete from Binary Tree ");

printf("\n\t 3. Inorder traversal of Binary tree");

printf("\n\t 4. Postorder traversal of Binary tree");

printf("\n\t 5. Preorder traversal of Binary tree");

printf("\n\t 6. Exit ");

printf("\n\t Enter choice : ");

scanf(" %d",&choice);

if(choice<1 || choice>7)

printf("\n Invalid choice - try again");

}while (choice<1 || choice>6);

switch(choice)

{

case 1:

printf("\n Enter new element: ");

scanf("%d", &item);

root= insert(root,item);

printf("\n root is %d",root->info);

printf("\n Inorder traversal of binary tree is : ");

inorder(root);

printf("\n Preorder traversal of binary tree is : ");

preorder(root);

printf("\n Postorder traversal of binary tree is : ");

postorder(root);

printf("\n");

break;

case 2:

printf("\n Enter the element to be deleted : ");

scanf(" %d",&item\_no);

root=delet(root,item\_no);

printf("\n Inorder traversal of binary tree is : ");

inorder(root);

printf("\n Preorder traversal of binary tree is : ");

preorder(root);

printf("\n Postorder traversal of binary tree is : ");

postorder(root);

printf("\n");

break;

case 3:

printf("\n Inorder traversal of binary tree is : ");

inorder(root);

break;

case 4:

printf("\n Postorder traversal of binary tree is : ");

postorder(root);

break;

case 5:

printf("\n Preorder traversal of binary tree is : ");

preorder(root);

break;

default:

printf("\n End of program ");

}

/\* end of switch \*/

}

while(choice !=6);

return(0);

}

struct tree \*insert(struct tree \*root, int x)

{

if(!root)

{

root=(struct tree\*)malloc(sizeof(struct tree));

root->info = x;

root->left = NULL;

root->right = NULL;

return(root);

}

if(root->info > x)

root->left = insert(root->left,x);

else

{

if(root->info < x)

root->right = insert(root->right,x);

}

return(root);

}

void inorder(struct tree \*root)

{

if(root != NULL)

{

inorder(root->left);

printf(" %d",root->info);

inorder(root->right);

}

return;

}

void postorder(struct tree \*root)

{

if(root != NULL)

{

postorder(root->left);

postorder(root->right);

printf(" %d",root->info);

}

return;

}

void preorder(struct tree \*root)

{

if(root != NULL)

{

printf(" %d",root->info);

preorder(root->left);

preorder(root->right);

}

return;

}

/\* FUNCTION TO DELETE A NODE FROM A BINARY TREE \*/

struct tree \*delet(struct tree \*ptr,int x)

{

struct tree \*p1,\*p2;

if(!ptr)

{

printf("\n Node not found ");

return(ptr);

}

else

{

if(ptr->info < x)

{

ptr->right = delet(ptr->right,x);

/\*return(ptr);\*/

}

else if (ptr->info >x)

{

ptr->left=delet(ptr->left,x);

return ptr;

}

else

/\* no. 2 else \*/

{

if(ptr->info == x)

/\* no. 2 if \*/ {

if(ptr->left == ptr->right)

/\*i.e., a leaf node\*/

{

free(ptr);

return(NULL);

}

else if(ptr->left==NULL)

/\* a right subtree \*/

{

p1=ptr->right;

free(ptr);

return p1;

}

else if(ptr->right==NULL)

/\* a left subtree \*/

{

p1=ptr->left;

free(ptr);

return p1;

}

else

{

p1=ptr->right;

p2=ptr->right;

while(p1->left != NULL)

p1=p1->left;

p1->left=ptr->left;

free(ptr);

return p2;

}

}

/\*end of no. 2 if \*/

}

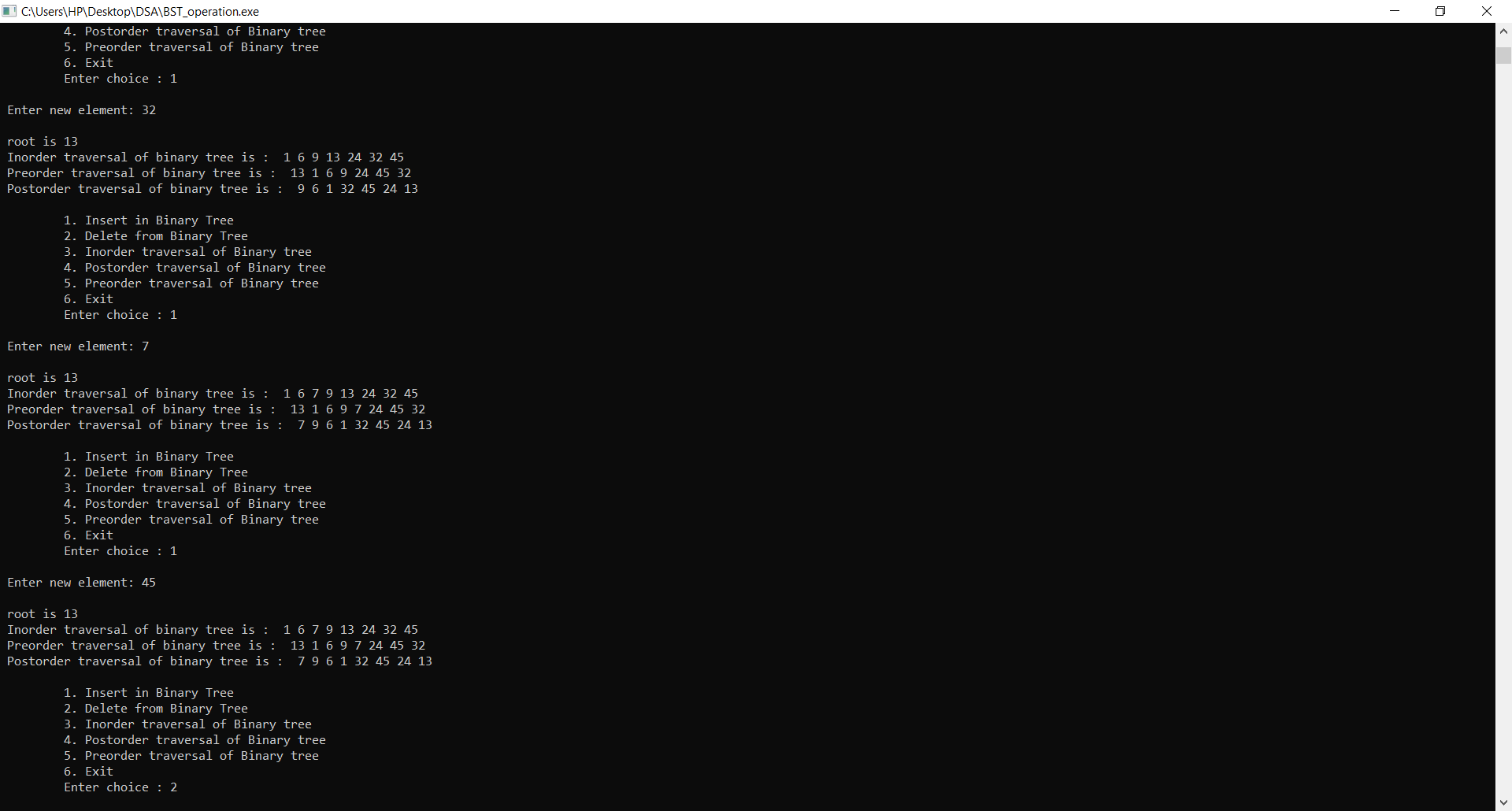
/\* end of no. 2 else \*/

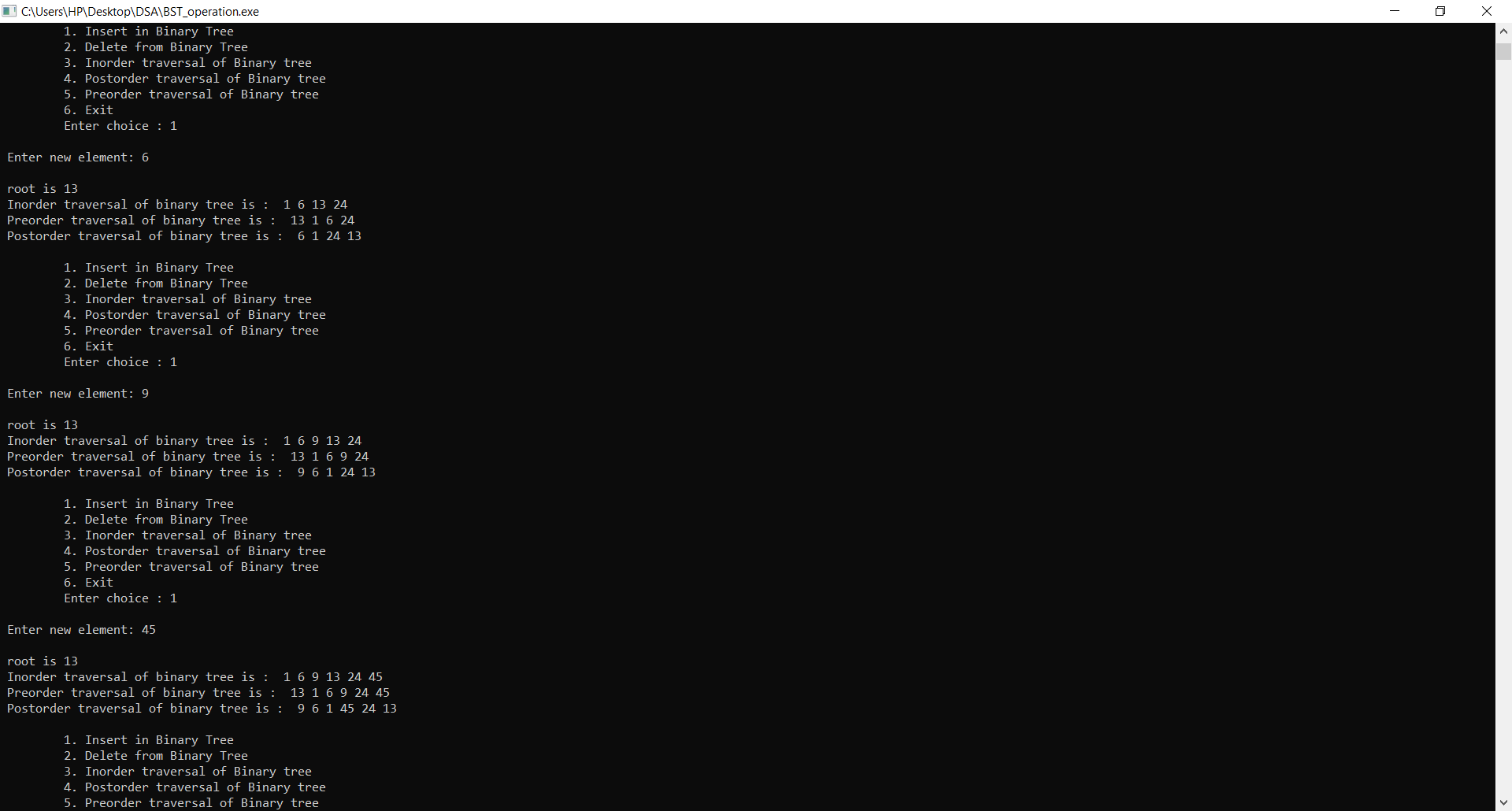
/\* check which path to search for a given no. \*/

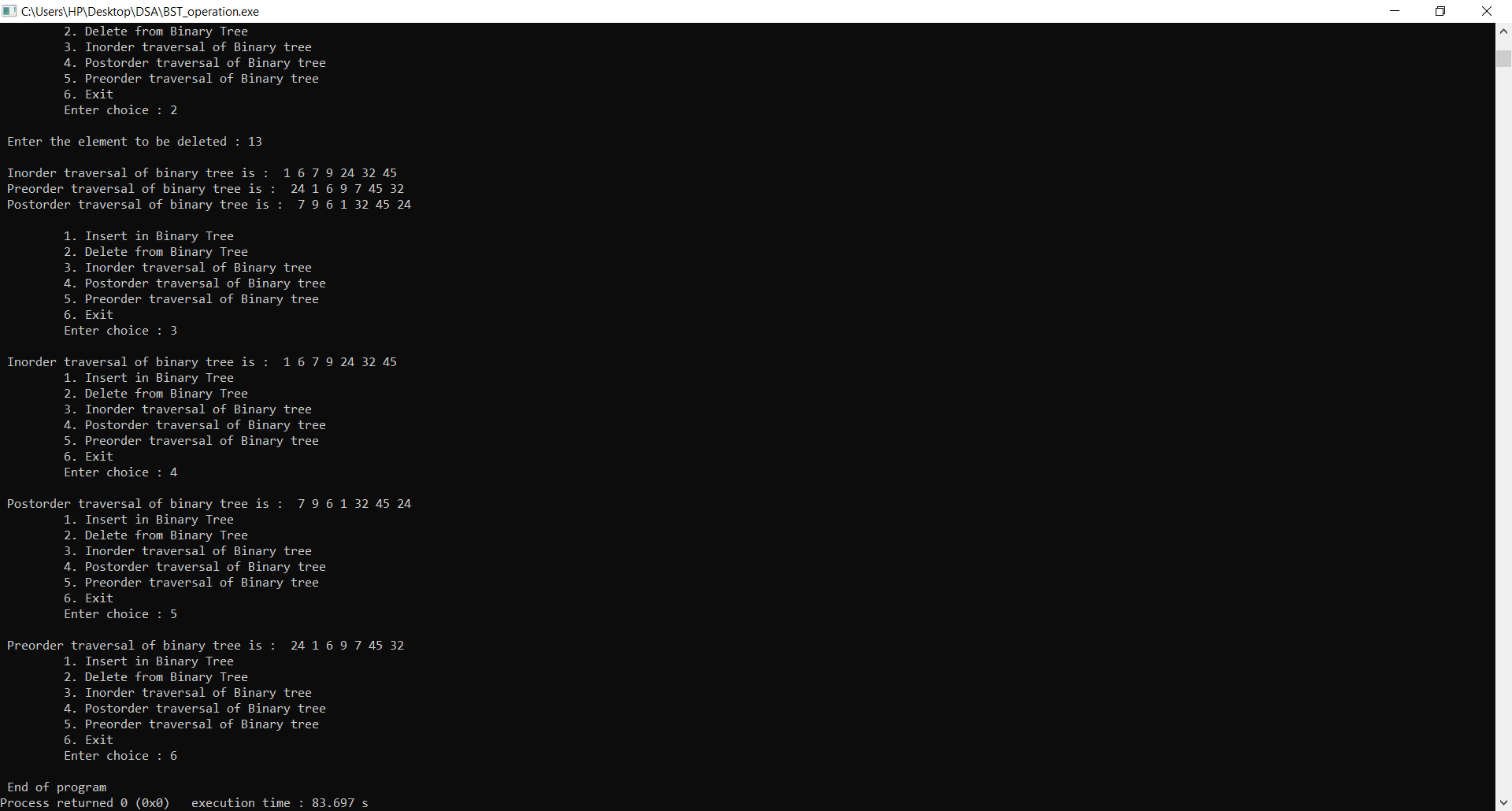
}

return(ptr);

}

****

****



**2. Implement C program to perform sorting of n numbers using heap sort technique.**

**PSEUDOCODE:**

void heapify(int arr[])

{

int i,n

n=arr[0]

for(i=n/2;i>=1;i--)

adjust(arr,i)

}

void adjust(int arr[],int i)

{

int j,temp,n,k=1

n=arr[0]

while(2\*i<=n && k==1)

{

j=2\*i

if(j+1<=n && arr[j+1] > arr[j])

j=j+1

if( arr[j] < arr[i])

k=0

else

{

temp=arr[i]

arr[i]=arr[j]

arr[j]=temp

i=j

}

}

}

int main()

{

int arr[100],n,temp,last

printf("Number of Elements you want in array: ")

scanf("%d",&n)

printf("Enter Elements in array:\n")

for(int i=1;i<=n;i++)

scanf("%d",&arr[i])

arr[0]=n

heapify(arr)

while(arr[0] > 1)

{

last=arr[0]

temp=arr[1]

arr[1]=arr[last]

arr[last]=temp

arr[0]--

adjust(arr,1)

}

printf("Sorted Array \n")

for(int i=1;i<=n;i++)

printf("%d ",arr[i])

}

**CODE:**

#include<stdio.h>

#include <conio.h>

void heapify(int arr[])

{

int i,n;

n=arr[0];

for(i=n/2;i>=1;i--)

adjust(arr,i);

}

void adjust(int arr[],int i)

{

int j,temp,n,k=1;

n=arr[0];

while(2\*i<=n && k==1)

{

j=2\*i;

if(j+1<=n && arr[j+1] > arr[j])

j=j+1;

if( arr[j] < arr[i])

k=0;

else

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

i=j;

}

}

}

int main()

{

int arr[100],n,temp,last;

printf("Number of Elements you want in array: ");

scanf("%d",&n);

printf("Enter Elements in array:\n");

for(int i=1;i<=n;i++)

scanf("%d",&arr[i]);

arr[0]=n;

heapify(arr);

while(arr[0] > 1)

{

last=arr[0];

temp=arr[1];

arr[1]=arr[last];

arr[last]=temp;

arr[0]--;

adjust(arr,1);

}

printf("Sorted Array \n");

for(int i=1;i<=n;i++)

printf("%d ",arr[i]);

getch();

return 0;

}

