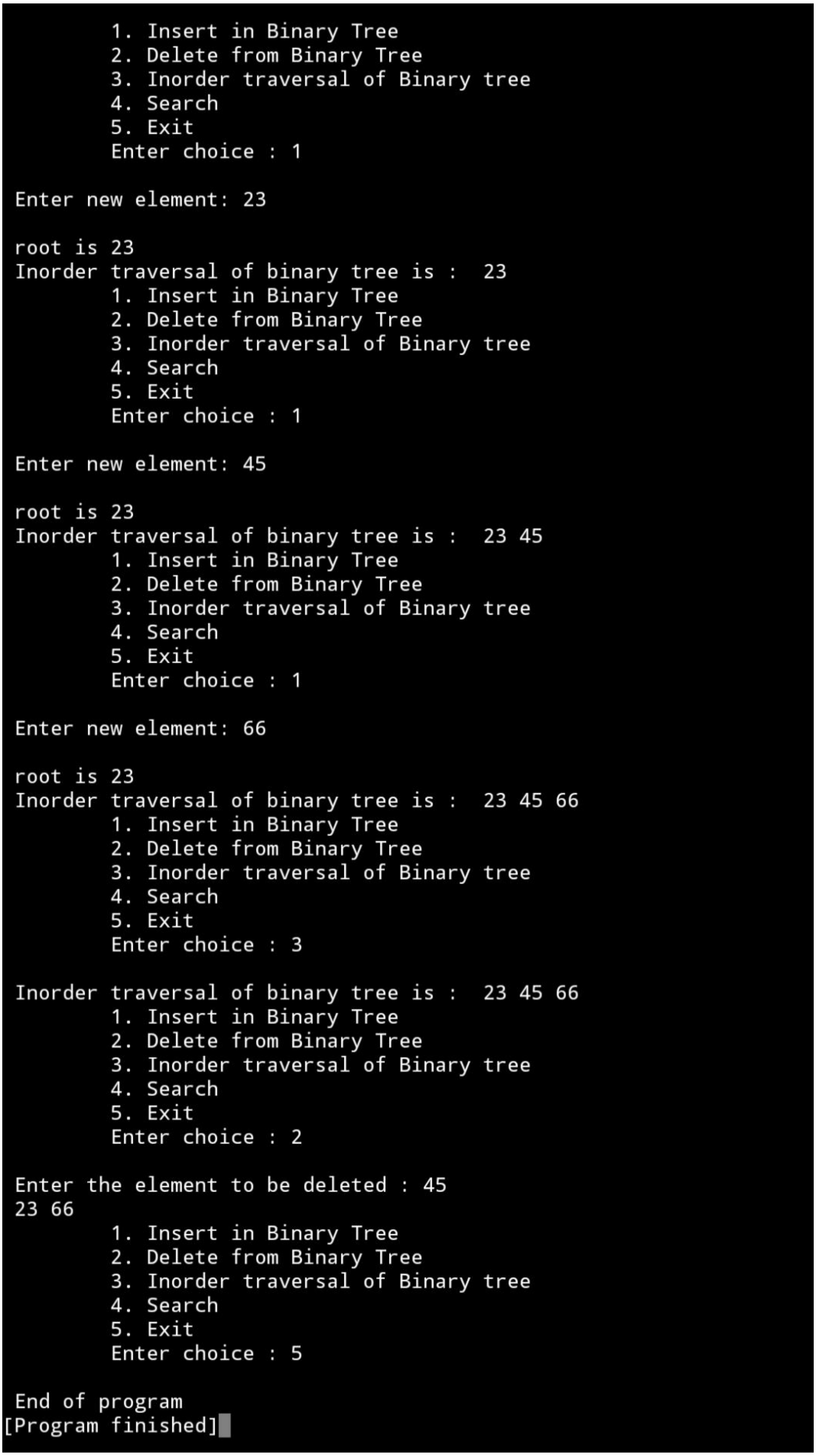
```
#include <stdlib.h>
#include<stdio.h>
struct tree {
 int info;
 struct tree *left;
 struct tree *right;
};
struct tree *insert(struct tree *,int);
void inorder(struct tree *);
void postorder(struct tree *);
void preorder(struct tree *);
struct tree *delet(struct tree *,int);
struct tree *search(struct tree *);
int main(void) {
 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. Search”);
 printf(“\n\t 5. Exit “);
 printf(“\n\t Enter choice : “);
 printf(“ %d”,&choice);
 if(choice<1 || choice>7)
 printf(“\n Invalid choice – try again”);
 }
 while (choice<1 || choice>7);
 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);
 break;
 case 2: printf(“\n Enter the element to be deleted : “);
 scanf(“ %d”,&item_no);
 root=delet(root,item_no);
 inorder(root);
 break;
 Case 3: printf(“\n Inorder traversal of binary tree is : “);
 inorder(root);
 break;
 Case 4: printf(“\n Search operation in binary tree “);
 root=search(root);
 break;
 default:
 printf(“\n End of program “);
 }
 }
 While(choice !=5);
 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;
}
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);
 } 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);
}
struct tree *search(struct tree *root) {
 int no,I,info;
 struct tree *ptr;
 ptr=root;
printf("\n Enter the element to be searched :");
 scanf(" %d",&no);
 fflush(stdin);
While(ptr) {
 If(no>ptr->info)
 Ptr=ptr->right; else if(no<ptr->info)
 Ptr=ptr->left; else
break;
  }
If(ptr) {
 printf(“\n Element %d which was searched is found and is = %d”,no,ptr->info);
 } else
printf(“\n Element %d does not exist in the binary tree”,no);
 return(root);
}
```

Output:-



```
         1. Insert in Binary Tree
         2. Delete from Binary Tree
         3. Inorder traversal of Binary tree
         4. Search
         5. Exit
         Enter choice : 1

 Enter new element: 23

 root is 23
 Inorder traversal of binary tree is :  23
         1. Insert in Binary Tree
         2. Delete from Binary Tree
         3. Inorder traversal of Binary tree
         4. Search
         5. Exit
         Enter choice : 1

 Enter new element: 45

 root is 23
 Inorder traversal of binary tree is :  23 45
         1. Insert in Binary Tree
         2. Delete from Binary Tree
         3. Inorder traversal of Binary tree
         4. Search
         5. Exit
         Enter choice : 1

 Enter new element: 66

 root is 23
 Inorder traversal of binary tree is :  23 45 66
         1. Insert in Binary Tree
         2. Delete from Binary Tree
         3. Inorder traversal of Binary tree
         4. Search
         5. Exit
         Enter choice : 3

 Inorder traversal of binary tree is :  23 45 66
         1. Insert in Binary Tree
         2. Delete from Binary Tree
         3. Inorder traversal of Binary tree
         4. Search
         5. Exit
         Enter choice : 2

 Enter the element to be deleted : 45
 23 66
         1. Insert in Binary Tree
         2. Delete from Binary Tree
         3. Inorder traversal of Binary tree
         4. Search
         5. Exit
         Enter choice : 5

 End of program
[Program finished]
```