Aim:- To study the implementation of Binary Search Tree .

Source Code:-

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node
{
  int data;
  struct node *left;
  struct node *right;
};

struct node *tree;
void create_tree (struct node *);
struct node *insertElement (struct node *, int);
void preorderTraversal (struct node *);
void inorderTraversal (struct node *);
void postorderTraversal (struct node *);
struct node *findSmallestElement (struct node *);
struct node *findLargestElement (struct node *);
struct node *deleteElement (struct node *, int);
int totalNodes (struct node *);

struct node *deleteTree (struct node *);
int
main ()
{
  int option, val;
  struct node *ptr;
  create_tree (tree);
  printf ("D10A_Atharva Chavan_9\n");
  printf ("\n 1. Insert an Element");
  printf ("\n 2. Preorder Traversal");
  printf ("\n 3. Inorder Traversal");
  printf ("\n 4. Postorder Traversal");
  printf ("\n 5. Find the smallest element");
  printf ("\n 6. Find the largest element");
  printf ("\n 7. Delete an element");
  printf ("\n 8. Count the total number of nodes");
  printf ("\n 9. Delete the tree");
  printf ("\n 10. Exit");

  do
   {
     printf ("\n Enter your option : ");
     scanf ("%d", &option);
     switch (option)
        {

case 1:
printf ("\n Enter the value of the new node : ");
        scanf ("%d", &val);
        tree = insertElement (tree, val);
        break;
```

```
          case 2:
printf ("\n The elements of the tree are : \n");
            preorderTraversal (tree);
            break;

          case 3:
printf ("\n The elements of the tree are : \n");
            inorderTraversal (tree);
            break;

          case 4:
printf ("\n The elements of the tree are : \n");
            postorderTraversal (tree);
            break;

          case 5:
ptr = findSmallestElement (tree);
            printf ("\n Smallest element is :%d", ptr->data);

break;

          case 6:
ptr = findLargestElement (tree);
            printf ("\n Largest element is : %d", ptr->data);
            break;

          case 7:
printf ("\n Enter the element to be deleted : ");
```

```
             scanf ("%d", &val);
             tree = deleteElement (tree, val);
             break;

           case 8:
printf ("\n Total no. of nodes = %d", totalNodes (tree));
             break;

           case 9:
             tree = deleteTree (tree);
             printf("\nThe tree is deleted");
             break;
}

}
  while (option != 10);
  printf("\nYou have exited the tree!");
  getch ();
  return 0;
}

void
create_tree (struct node *tree)
{
tree = NULL;
}

struct node *
```

```
insertElement (struct node *tree, int val)
{
  struct node *ptr, *nodeptr, *parentptr;
  ptr = (struct node *) malloc (sizeof (struct node));
  ptr->data = val;
  ptr->left = NULL;
  ptr->right = NULL;
  if (tree == NULL)

    {
      tree = ptr;
      tree->left = NULL;
      tree->right = NULL;
}

  else
    {
parentptr = NULL;
      nodeptr = tree;
      while (nodeptr != NULL)
           {

parentptr = nodeptr;
             if (val < nodeptr->data)
               nodeptr = nodeptr->left;
             else
               nodeptr = nodeptr->right;
}
if (val < parentptr->data)
           parentptr->left = ptr;
      else
           parentptr->right = ptr;
}
  return tree;
}

void
preorderTraversal (struct node *tree)
{
  if (tree != NULL)
    {
      printf ("%d\t", tree->data);
      preorderTraversal (tree->left);
      preorderTraversal (tree->right);
}
}

void
inorderTraversal (struct node *tree)
{
if (tree != NULL)
    {
inorderTraversal (tree->left);
      printf ("%d\t", tree->data);
      inorderTraversal (tree->right);
```

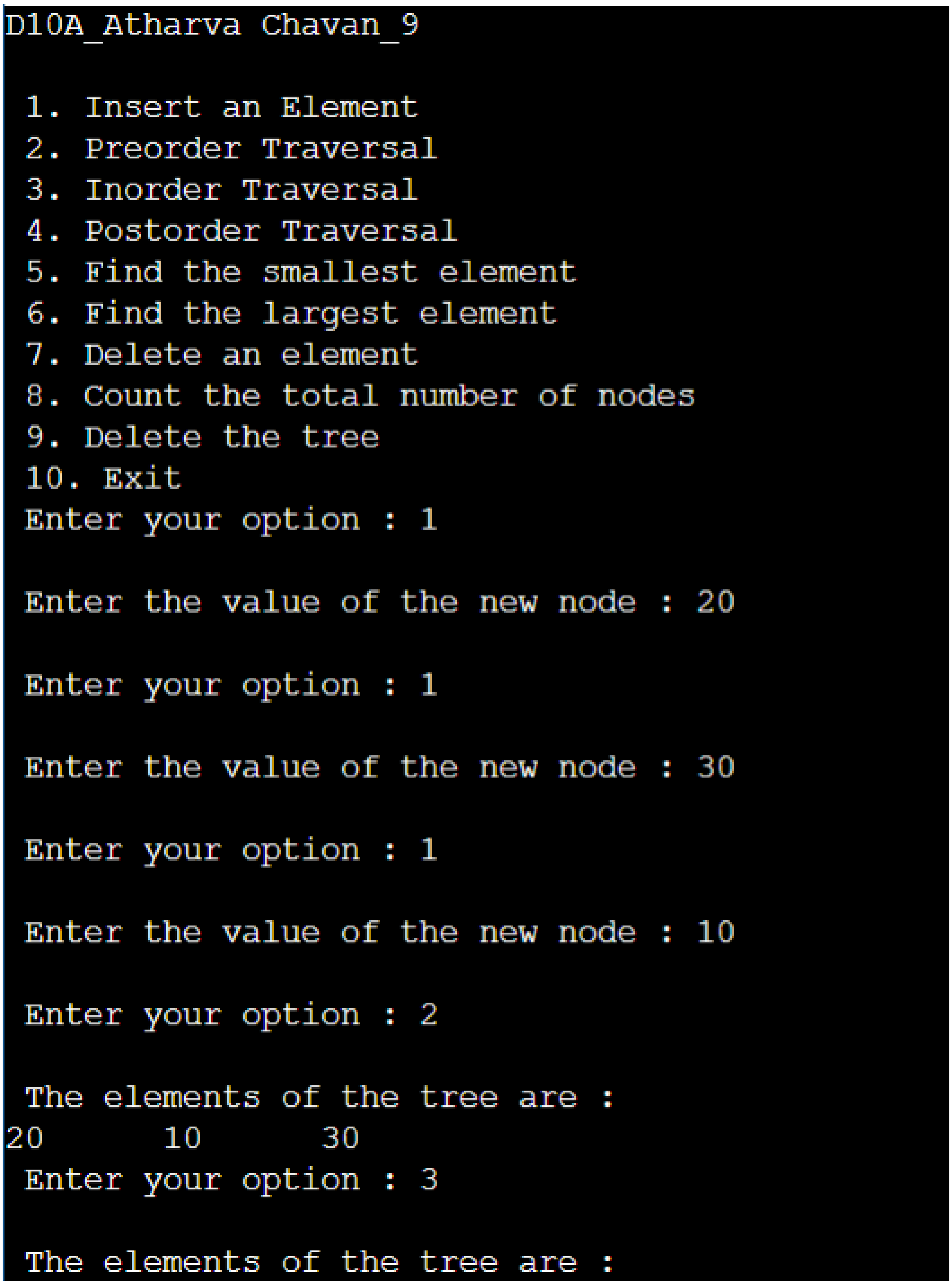
```
}
}

void
postorderTraversal (struct node *tree)
{
if (tree != NULL)
   {

postorderTraversal (tree->left);
      postorderTraversal (tree->right);
      printf ("%d\t", tree->data);
}
}

struct node *
findSmallestElement (struct node *tree)
{
  if ((tree == NULL) || (tree->left == NULL))
    return tree;
  else
    return findSmallestElement (tree->left);
}

struct node *
findLargestElement (struct node *tree)
{
if ((tree == NULL) || (tree->right == NULL))
```

```
    return tree;
  else
    return findLargestElement (tree->right);
}

struct node *
deleteElement (struct node *tree, int val)
{
  struct node *cur, *parent, *suc, *psuc, *ptr;
  if (tree->left == NULL)
    {
      printf ("\n The tree is empty ");
      return (tree);
}

parent = tree;
  cur = tree->left;

while (cur != NULL && val != cur->data)
    {

parent = cur;
      cur = (val < cur->data) ? cur->left : cur->right;


}

if (cur == NULL)
    {
      printf ("\n The value to be deleted is not present in the tree");
      return (tree);
}

if (cur->left == NULL)
    ptr = cur->right;
  else if (cur->right == NULL)
    ptr = cur->left;
  else

    {
psuc = cur;
      cur = cur->left;
      while (suc->left != NULL)
          {

psuc = suc;
             suc = suc->left;
}

if (cur == psuc)
          {
             suc->left = cur->right;
}
else                {

suc->left = cur->left;
             psuc->left = suc->right;
```

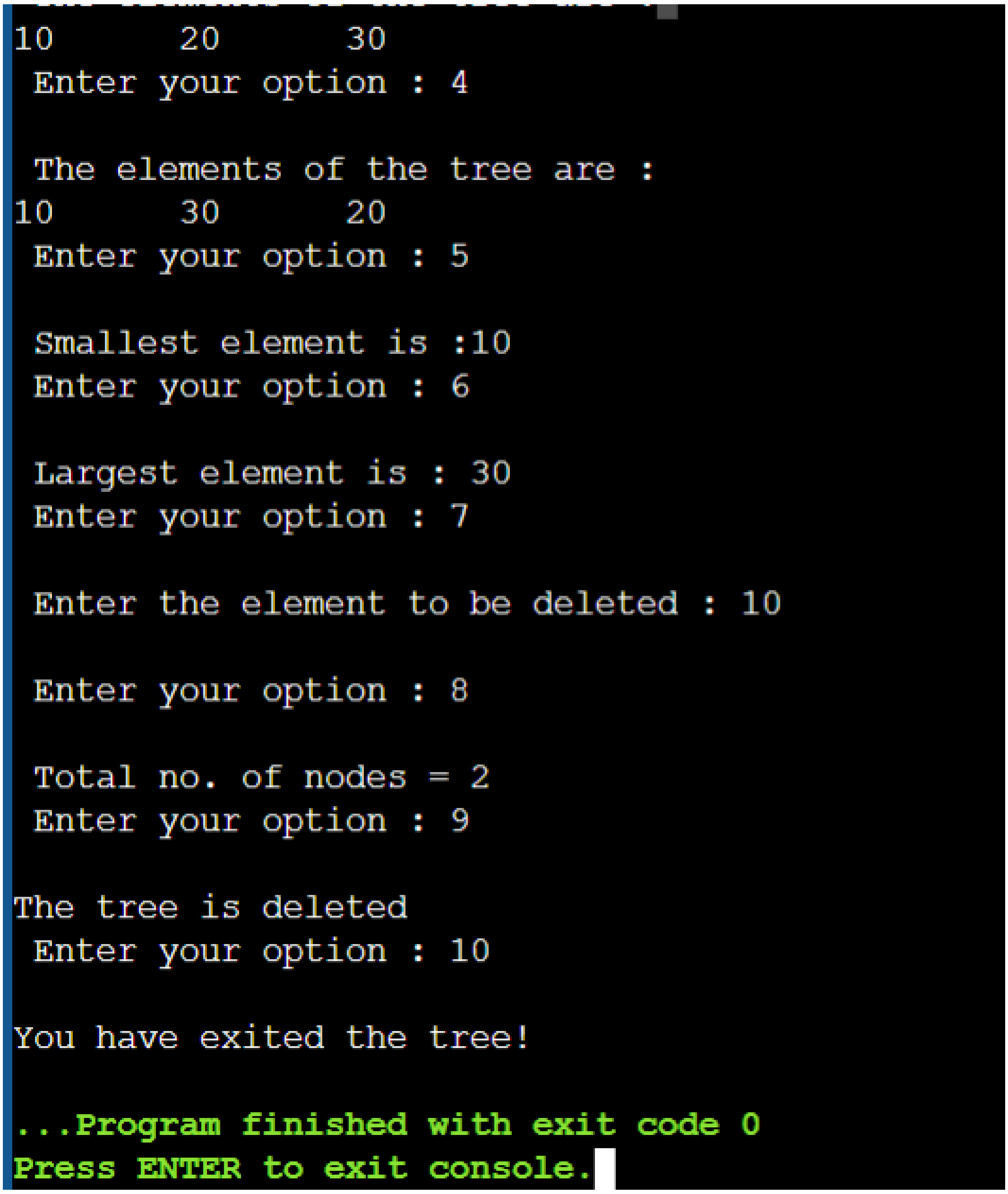
```
            suc->right = cur->right;

}
      ptr = suc;

}

if (parent->left == cur)
    parent->left = ptr;
  else
    parent->right = ptr;
  free (cur);
  return tree;
}

int
totalNodes (struct node *tree)
{
if (tree == NULL)
    return 0;
  else
    return (totalNodes (tree->left) + totalNodes (tree->right) + 1);
}

struct node *
deleteTree (struct node *tree)
{
if (tree != NULL)
```

```
   {
     deleteTree (tree->left);
     deleteTree (tree->right);
     free (tree);
}
}
```

Output:-



```
D10A_Atharva Chavan_9

 1. Insert an Element
 2. Preorder Traversal
 3. Inorder Traversal
 4. Postorder Traversal
 5. Find the smallest element
 6. Find the largest element
 7. Delete an element
 8. Count the total number of nodes
 9. Delete the tree
 10. Exit
 Enter your option : 1

 Enter the value of the new node : 20

 Enter your option : 1

 Enter the value of the new node : 30

 Enter your option : 1

 Enter the value of the new node : 10

 Enter your option : 2

 The elements of the tree are :
20      10      30
 Enter your option : 3

 The elements of the tree are :
```



```
10       20       30
 Enter your option : 4

 The elements of the tree are :
10       30       20
 Enter your option : 5

 Smallest element is :10
 Enter your option : 6

 Largest element is : 30
 Enter your option : 7

 Enter the element to be deleted : 10

 Enter your option : 8

 Total no. of nodes = 2
 Enter your option : 9

The tree is deleted
 Enter your option : 10

You have exited the tree!

...Program finished with exit code 0
Press ENTER to exit console.
```