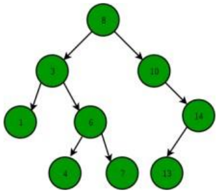
EXNO:5 **IMPLEMENTATION BINARY TREE AND OPERATIONS OF BINARY TREES**

AIM:

To write a C program Implementation Binary Tree And Operations Of Binary Trees

DESCRIPTION:

A binary tree is a tree data structure where each node has up to two child nodes, creating the branches of the tree. The two children are usually called the left and right nodes. Parent nodes are nodes with children, while child nodes may include references to their parents.



ALGORITHM

1.Start from root.

2. Compare the inserting element with root, if less than root, then recurse for left, else recurse for right.

3. If element to search is found anywhere, return true, else return false

PROGRAM

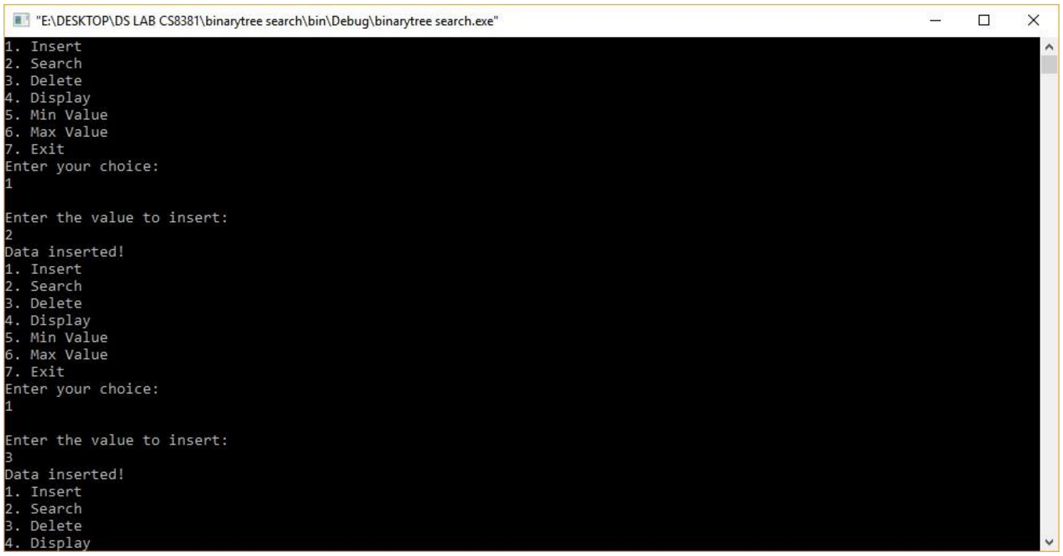
```
#include<stdio.h>
#include<stdlib.h>

struct tree
{
      int data;
      struct tree
*left;
      struct tree *right;
} *root = NULL, *node = NULL, *temp = NULL;

struct tree* insert(int key,struct tree *leaf)
{     if(leaf == 0) {         struct tree
*temp;
            temp = (struct tree *)malloc(sizeof(struct tree));
            temp->data = key;
      temp->left = 0;         temp-
>right = 0;
            printf("Data inserted!\n");
      return temp;
      }
      else {
            if(key < leaf->data)                leaf-
>left = insert(key,leaf->left);           else
                  leaf->right = insert(key,leaf->right);
      }
      return leaf;
}

struct tree* search(int key,struct tree *leaf) {
      if(leaf != NULL) {      if(key ==
      leaf->data) {           printf("Data
      found!\n");
                  return leaf;
            }
```

```
else {
      if(key < leaf->data)
return search(key,leaf->left);
      else
```

```
                        return search(key,leaf->right);
            }
      }


else {
            printf("Data not found!\n");return NULL;


}
}
struct tree* minvalue(struct tree *node) {
      if(node == NULL)
            return NULL;

      if(node->left)
      return minvalue(node->left);
      else
      return node;
}

/* Function for find maximum value from the
Tree */ struct tree* maxvalue(struct tree
*node) {    if(node == NULL)
            return NULL;


      if(node->right)         return
maxvalue(node->right);
      else
      return node;
}
void preorder(struct tree *leaf) {
      if(leaf == NULL)
            return;
      printf("%d\n",leaf->data);
      preorder(leaf->left);
      preorder(leaf->right);
}
  void inorder(struct tree
*leaf) {    if(leaf == NULL)
            return;
      preorder(leaf->left);   printf("%d\n",leaf-
>data);
      preorder(leaf->right);
}

void postorder(struct tree *leaf) {
      if(leaf == NULL)
            return;
      preorder(leaf->left);   preorder(leaf-
>right);    printf("%d\n",leaf->data);
```

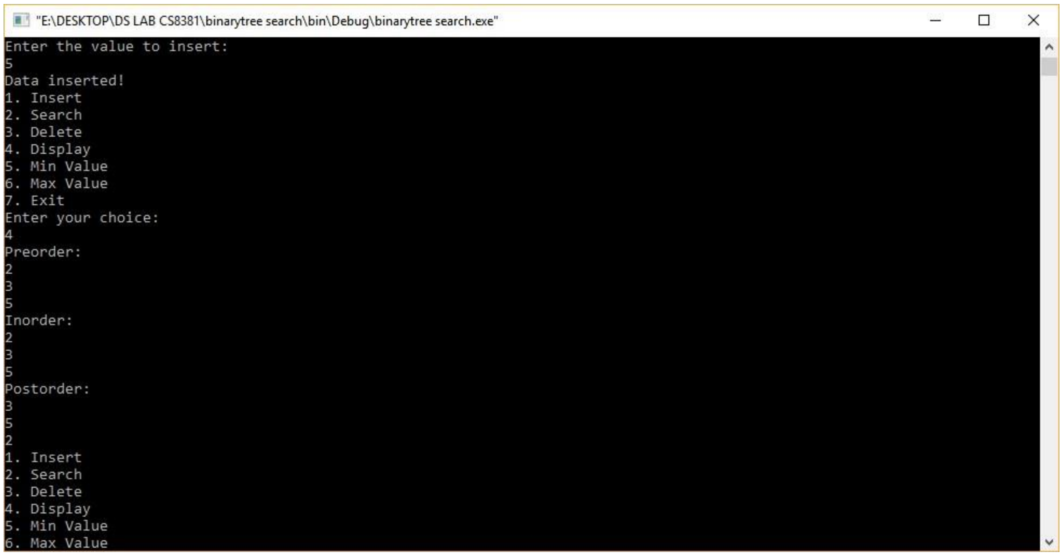
```
}

struct tree* delete(struct tree *leaf, int
key) {      if(leaf == NULL)
      printf("Element Not Found!\n");     else
if(key < leaf->data)          leaf->left =
delete(leaf->left, key); else if(key > leaf-
>data)      leaf->right = delete(leaf->right,
key);
      else {
            if(leaf->right && leaf->left) {


                  temp = minvalue(leaf->right);
            leaf->data = temp->data;


                   leaf->right = delete(leaf->right,temp->data);



}
            else {

                  temp = leaf;
                  if(leaf->left == NULL)
                        leaf = leaf->right;
                  else if(leaf->right == NULL)
                        leaf = leaf->left;
                  free(temp);
                  printf("Data delete successfully!\n");
            }
      }
}
 int
main()
{
      int key, choice;        while(choice != 7) {          printf("1.
Insert\n2. Search\n3. Delete\n4. Display\n5. Min Value\n6. Max Value\n7.
Exit\n");
            printf("Enter your choice:\n");
      scanf("%d", &choice);
            switch(choice) {
                  case 1:
                        printf("\nEnter the value to
insert:\n");                        scanf("%d", &key);
                        root = insert(key, root);
                        break;
                  case 2:
                        printf("\nEnter the value to
search:\n");                        scanf("%d", &key);
                        search(key,root);
            break;
         case 3:
            printf("\nEnter the value to
delete:\n");      scanf("%d", &key);
         delete(root,key);
            break;
         case 4:
            printf("Preorder:\n");
         preorder(root);
   printf("Inorder:\n");
   inorder(root);
   printf("Postorder:\n");
            postorder(root);
            break;
         case 5:
            if(minvalue(root) == NULL)
            printf("Tree is empty!\n");
            else
               printf("Minimum value is %d\n", minvalue(root)-
>data);
            break;
         case 6:
            if(maxvalue(root) == NULL)
               printf("Tree is empty!\n");
            else
               printf("Maximum value is %d\n", maxvalue(root)-
>data);
            break;
         case 7:
            printf("Bye
   Bye!\n");  exit(0);
            break;
      default:
            printf("Invalid choice!\n");
      }
   }
   retu
   rn
   0; }
```

OUT
PUT



"E:\DESKTOP\DS LAB CS8381\binarytree search\bin\Debug\binarytree search.exe"

```
1. Insert
2. Search
3. Delete
4. Display
5. Min Value
6. Max Value
7. Exit
Enter your choice:
1

Enter the value to insert:
2
Data inserted!
1. Insert
2. Search
3. Delete
4. Display
5. Min Value
6. Max Value
7. Exit
Enter your choice:
1

Enter the value to insert:
3
Data inserted!
1. Insert
2. Search
3. Delete
4. Display
```



"E:\DESKTOP\DS LAB CS8381\binarytree search\bin\Debug\binarytree search.exe"

```
Enter the value to insert:
5
Data inserted!
1. Insert
2. Search
3. Delete
4. Display
5. Min Value
6. Max Value
7. Exit
Enter your choice:
4
Preorder:
2
3
5
Inorder:
2
3
5
Postorder:
3
5
2
1. Insert
2. Search
3. Delete
4. Display
5. Min Value
6. Max Value
```

Result:

Thus the program in C is implementated Binary Tree and Operations of Binary Trees.