```
//By Ritwik Chandra Pandey
//Threaded Binary Tree
//On 27th August 2021

#include <stdio.h>
#include <stdlib.h>
#define true 1
#define false 0
#define MAX_VALUE 65536


struct ThreadedBinaryNode {
    int data;
    struct ThreadedBinaryNode * left, *right;
    //1 - indicates a thread, 0 - indicates not a thread
    int leftThread, rightThread;
};
typedef struct ThreadedBinaryNode * TBNODE;
TBNODE root =  NULL;
void insert(int ele) {
    TBNODE p = root;
    for(;;) {
       if(p->data = ele){
           if(p->rightThread)
             break;
           p = p->right;
```

```
        }else if( p->data=ele){
            if(p->leftThread)
                break;
            p = p->left;
        }else{
            printf("Duplicates are not allowed.\n");
            return;
        }
    }
    TBNODE tmp = (TBNODE)malloc(sizeof(struct ThreadedBinaryNode));
    tmp-> data = ele;
    tmp->rightThread = tmp->leftThread = true;
    if (p->data <ele){
        tmp->right = p->right;
        tmp->left = p;
        p-> right = tmp;
        p->rightThread = false;
}else{

        tmp->right = p;
        tmp->left = p->left;
        p->left = tmp;
        p->leftThread = false;
    }
}
void delete(int ele) {
```

```
    TBNODE dest = root->left;
    TBNODE p = root;
    while(1){
      if (dest->data < ele){
          if(dest->rightThread == true){
              printf("Cannot find %d in the threaded binary tree.\n",ele);
              return;
          }
          p=dest;
          dest = dest->right;
      }else if(dest->data>ele){
          if(dest->leftThread == true){

          printf("Cannot find %d in the threaded binary tree.\n",ele);
          return;
      }
      p =dest;
      dest = dest->left;
    }else{
      break;
    }
}
TBNODE target =dest;
if(dest->rightThread == false && dest->leftThread == false ){
    p=dest;
    target = dest->left;
```

```
    while(target->rightThread == false){
      p=target;
      target = target->right;

    }
    dest->data = target->data;
}
if(p->data > target->data){
     if (target->rightThread == true && target-> leftThread == true ){
       p->left = target->left;
       p->leftThread = true;
     }
     else if(target->rightThread == true){
       TBNODE largest = target-> left;
       while(largest->rightThread == false){
           largest = largest-> right;
       }
       largest->right = p;
       p->left = target-> left;
     }
     else{
       TBNODE smallest = target->right;
       while(smallest->leftThread ==   false ){
           smallest = smallest->left;
       }
       smallest->left = target ->left;
```

```
        p->left = target->right;
     }
}else{
    if (target-> rightThread == true && target->leftThread == true){
        p->right = target->right;
        p->rightThread=true;

    }else if(target->rightThread == true ){
        TBNODE largest = target->left;
        while(largest->rightThread ==  false  ){
            largest = largest->right;
        }
        largest->right = target->right;
        p->right = target->left;
    }else{
        TBNODE smallest = target->right;

              while (smallest->leftThread == false  ){
               smallest = smallest->left;
            }
            smallest->left = p;
            p->right = target->right;
        }
    }
}
int search(int ele){
```

```
    TBNODE temp = root->left;
    while(1){
      if(temp->data < ele){
            if(temp->rightThread  == true ){
               return false;
                    }
      temp= temp->right;

    }else if(temp->data>ele){
      if(temp->leftThread  == true   ){
         return false;
      }
      temp= temp->left;
    }else{
      return true;
    }
}
}
void traverse() {
    TBNODE tmp = root, p;
     for(;;){
      p = tmp;
      tmp = tmp->right;
      if(!p->rightThread){
         while(!tmp->leftThread){
            tmp = tmp->left;
```

```
        }
      }
      if(tmp==root)
        break;
      printf("%d ", tmp->data);
    }
    printf("\n");}


int main(){
  int ele, op, pos;
  root = (TBNODE)malloc(sizeof(struct ThreadedBinaryNode));
  root->right = root->left = root;
  root->leftThread = true;
  root->data = MAX_VALUE;
  while(1)
  {
    printf("1.Insert 2.Delete 3.Search 4.Traversal 5.Exit\n");
    printf("Enter your option : ");
    scanf("%d", &op);
    switch(op) {
      case 1: printf("Enter an element to be inserted : ");
        scanf("%d", &ele);
        insert(ele);
        break;
      case 2: printf("Enter the element to be deleted : ");
```

```
        scanf("%d", &ele);
        delete(ele);
        break;
      case 3: printf("Enter the element to be searched : ");
        scanf("%d", &ele);
        pos = search(ele);
        if(pos)
          printf("Element found in threaded binary tree.\n");
        else
          printf("Element not found in threaded binary tree.\n");
        pos=0;
        break;
      case 4:
        if(root->right == root && root->left == root){
          printf("Threaded binary tree is empty.\n");
        }
        else {
          printf("Elements of the threaded binary tree :");
          traverse(root);
        }
        break;

       case 5: exit(0);
  }
  }
}
```