Experiment no. 8

Objective: Implementation of Binary search tree

Code:

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

struct btnode
{
   int value;
   struct btnode *l;
   struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;

void delete1();
void insert();
void delete();
void create();
void search(struct btnode *t);
void display(struct btnode *t);
void search1(struct btnode *t,int data);

int flag = 1;

void main()
{
```

```
int ch;

printf("\n**********MENU**********\n");
printf("1.Insert node into tree\n");
printf("2.Delete node from the tree\n");
printf("3.Display\n");
printf("4.Search\n");
printf("5.Exit\n");
while(1)
{
    printf("\nEnter your choice: ");
    scanf("%d", &ch);
    switch (ch)
    {
    case 1:
        insert();
        break;
    case 2:
        delete();
        break;
    case 3:
        display(root);
        break;
    case 4:
        search(root);
    case 5:
        exit(0);
        default :
            printf("Please enter correct choice:");
            break;
        }
    }
}
void insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}
void create()
{
    int data;

    printf("Enter node to be inserted:");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
}
void search(struct btnode *t)
{
    int data;
    printf("Enter Element to Search:");
    scanf("%d",&data);
    if ((temp->value > t->value) && (t->r != NULL))search(t->r);
    else if ((temp->value > t->value) && (t->r == NULL))t->r = temp;
    else if ((temp->value < t->value) && (t->l != NULL))search(t->l);
    else if ((temp->value < t->value) && (t->l == NULL))t->l = temp;
    printf("Element is present");
}
void delete()
{
    int data;

    if (root == NULL)
    {
        printf("No elements in a tree to delete");
        return;
    }
    printf("Enter the data to be deleted : ");
    scanf("%d", &data);
    t1 = root;
    t2 = root;
    search1(root, data);
}
void display(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display");
        return;
    }
    printf("%d -> ", t->value);
    if (t->l != NULL)
        display(t->l);
    if (t->r != NULL)
        display(t->r);
}
void search1(struct btnode *t, int data)
{
    if ((data>t->value))
    {
        t1 = t;
        search1(t->r, data);
    }
    else if ((data < t->value))
    {
        t1 = t;
        search1(t->l, data);
    }
    else if ((data==t->value))
    {
        delete1(t);
    }
}
void delete1(struct btnode *t)
```

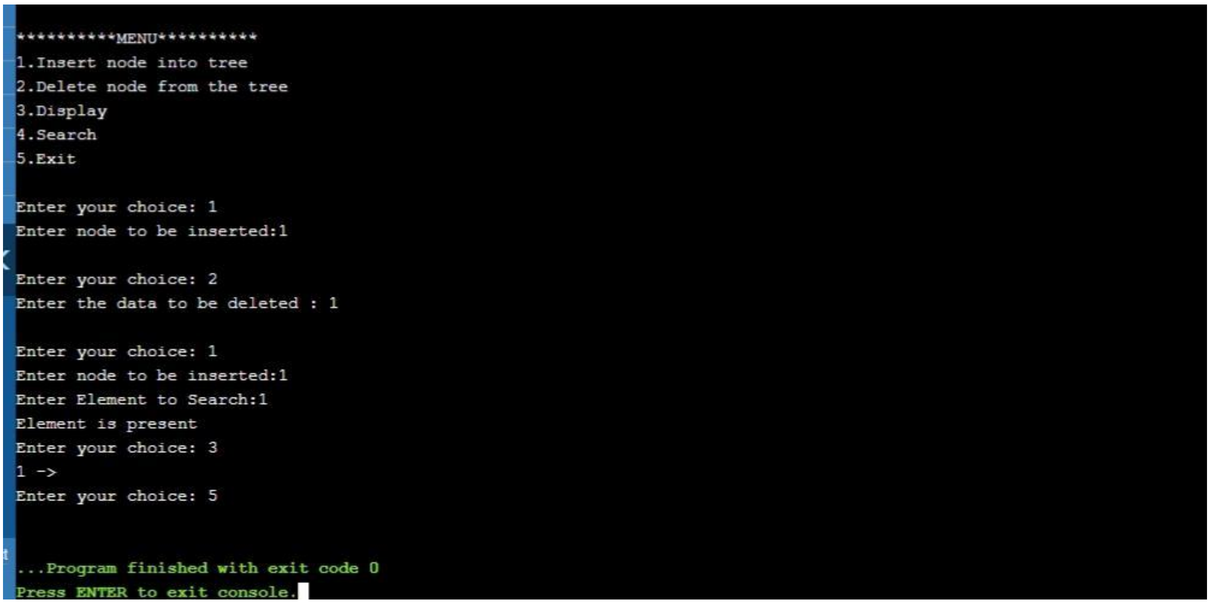
```
{
    int k;
    if ((t->l == NULL) && (t->r == NULL))
    {
        if (t1->l == t)
        {
            t1->l = NULL;
        }
        else
        {
            t1->r = NULL;
        }
        t = NULL;
        free(t);
        return;
    }
    else if ((t->r == NULL))
    {
        if (t1 == t)
        {
            root = t->l;
            t1 = root;
        }
        else if (t1->l == t)
        {
            t1->l = t->l;
        }
        else
        {
            t1->r = t->l;
        }
        t = NULL;
        free(t);
        return;
    }
    else if (t->l == NULL)
    {
        if (t1 == t)
        {
            root = t->r;
            t1 = root;
        }
        else if (t1->r == t)
            t1->r = t->r;
        else
            t1->l=t->r;
        t == NULL;
        free(t);
        return;
    }
    else if ((t->l != NULL) && (t->r != NULL))
    {
        t2 = root;
```

```
        if (t->r != NULL)
        {
            k = smallest(t->r);
            flag = 1;
        }
        else
        {
            k =largest(t->l);
            flag = 2;
        }
        search1(root, k);
        t->value = k;
    }

}
int smallest(struct btnode *t)
{
    t2 = t;
    if (t->l != NULL)
    {
        t2 = t;
        return(smallest(t->l));
    }
    else
        return (t->value);
}
```

```
int largest(struct btnode *t)
{
    if (t->r != NULL)
    {
        t2 = t;
        return(largest(t->r));
    }
    else
        return(t->value);
}
```

Output:



```
**********MENU**********
1.Insert node into tree
2.Delete node from the tree
3.Display
4.Search
5.Exit

Enter your choice: 1
Enter node to be inserted:1

Enter your choice: 2
Enter the data to be deleted : 1

Enter your choice: 1
Enter node to be inserted:1
Enter Element to Search:1
Element is present
Enter your choice: 3
1 ->
Enter your choice: 5


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

Presented by: Gelle Hruthesh reddy (20BCB7031)