

# **CSE 2003: Lab Assignment #7**

Due on Thursday, March 16, 2017

*Prof. Shaik Naseera 2:00pm*

**Jacob John**

## Contents

|           |   |
|-----------|---|
| Problem 1 | 3 |
|-----------|---|

## Problem 1

Implement a binary search tree using C.

Listing 1: Binary search tree using C

```
/*Recursive operations in Binary Search Tree*/
#include<stdio.h>
#include<stdlib.h>
struct node
5 {
    struct node *lchild;
    int info;
    struct node *rchild;
};
10 struct node *search(struct node *ptr,int skey);
struct node *insert(struct node *ptr,int ikey);
struct node *del(struct node *ptr,int dkey);
struct node *Min(struct node *ptr);
struct node *Max(struct node *ptr);
15 void preorder(struct node *ptr);
void inorder(struct node *ptr);
void postorder(struct node *ptr);
int height(struct node *ptr);

20 int main()
{
    struct node *root=NULL,*ptr;
    int choice,k;
    while(1)
25 {
        printf("\n");
        printf("1.Search\n");
        printf("2.Insert\n");
        printf("3.Delete\n");
30 printf("4.Preorder Traversal\n");
        printf("5.Inorder Traversal\n");
        printf("6.Postorder Traversal\n");
        printf("7.Height of tree\n");
        printf("8.Find minimum and maximum\n");
35 printf("9.Quit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
40         case 1:
            printf("Enter the key to be searched: ");
            scanf("%d",&k);
            ptr = search(root,k);
            if(ptr==NULL)
45             printf("Key not present\n");
            else
                printf("Key present\n");
            break;
```

```
50      case 2:
        printf("Enter the key to be inserted: ");
        scanf("%d",&k);
        root = insert(root,k);
        break;

55      case 3:
        printf("Enter the key to be deleted: ");
        scanf("%d",&k);
        root = del(root,k);
        break;

60      case 4:
        preorder(root);
        break;

65      case 5:
        inorder(root);
        break;

70      case 6:
        postorder(root);
        break;

75      case 7:
        printf("Height of the tree is %d\n",height(root));
        break;

        case 8:
        ptr= Min(root);
        if(ptr!=NULL)
            printf("Minimum key is %d\n",ptr->info);
        ptr = Max(root);
        if(ptr!=NULL)
            printf("Maximum key is %d\n",ptr->info);
85      break;

        case 9:
            exit(1);

90      default:
            printf("Wrong choice\n");
            }/*End of switch*/
        }/*End of while*/
    }/*End of main()*/

95 struct node *search(struct node *ptr,int skey)
{
    if(ptr == NULL)
    {
100        printf("key not found\n");
        return NULL;
    }
```

```

    }
    else if(skey < ptr->info) /*Search in left subtree*/
        return search(ptr->lchild, skey);
105    else if(skey > ptr->info) /*Search in right subtree*/
        return search(ptr->rchild, skey);
    else /*skey found*/
        return ptr;
} /*End of search() */

110 struct node *insert(struct node *ptr, int ikey)
{
    if(ptr==NULL) /*Base Case*/
    {
115        ptr = (struct node *)malloc(sizeof(struct node));
        ptr->info = ikey;
        ptr->lchild = NULL;
        ptr->rchild = NULL;
    }
120    else if(ikey < ptr->info) /*Insertion in left subtree*/
        ptr->lchild = insert(ptr->lchild, ikey);
    else if(ikey > ptr->info) /*Insertion in right subtree*/
        ptr->rchild = insert(ptr->rchild, ikey);
    else
125        printf("Duplicate key\n"); /*Base Case*/
    return ptr;
} /*End of insert*/

struct node *del(struct node *ptr, int dkey)
130 {
    struct node *tmp, *succ;
    if(ptr==NULL)
    {
        printf("dkey not found\n");
135        return ptr;
    }
    if(dkey < ptr->info) /*Deletion from left subtree*/
        ptr->lchild = del(ptr->lchild, dkey);
    else if(dkey > ptr->info) /*Deletion from right subtree*/
140        ptr->rchild = del(ptr->rchild, dkey);
    else
    {
        if(ptr->lchild!=NULL && ptr->rchild!=NULL) /*2 children*/
        {
145            succ = ptr->rchild;
            while(succ->lchild)
                succ = succ->lchild;
            ptr->info = succ->info;
            ptr->rchild = del(ptr->rchild, succ->info);
150        }
        else
        {
            tmp = ptr;
            if(ptr->lchild!=NULL) /*only left child*/

```

```
155         ptr = ptr->lchild;
            else if(ptr->rchild != NULL) /*only right child*/
                ptr = ptr->rchild;
            else /*no child*/
                ptr = NULL;
160         free(tmp);
    }
}
return ptr;
} /*End of del() */

165 struct node *Min(struct node *ptr)
{
    if(ptr==NULL)
        return NULL;
170     else if(ptr->lchild==NULL)
        return ptr;
    else
        return Min(ptr->lchild);
} /*End of Min() */

175 struct node *Max(struct node *ptr)
{
    if(ptr == NULL)
        return NULL;
180     else if(ptr->rchild==NULL)
        return ptr;
    else
        return Max(ptr->rchild);
} /*End of Max() */

185 int height(struct node *ptr)
{
    int h_left,h_right;
    if(ptr==NULL) /*Base Case*/
190         return 0;
    h_left = height(ptr->lchild);
    h_right = height(ptr->rchild);
    if(h_left > h_right)
        return 1 + h_left;
195     else
        return 1 + h_right;
} /*End of height() */

void preorder(struct node *ptr)
200 {
    if(ptr==NULL) /*Base Case*/
        return;
    printf("%d ",ptr->info);
    preorder(ptr->lchild);
205     preorder(ptr->rchild);
} /*End of preorder() */
```

```

void inorder(struct node *ptr)
{
210     if(ptr==NULL) /*Base Case*/
        return;
    inorder(ptr->lchild);
    printf("%d ",ptr->info);
    inorder(ptr->rchild);
215 }/*End of inorder()*/

void postorder(struct node *ptr)
{
220     if(ptr==NULL) /*Base Case*/
        return;
    postorder(ptr->lchild);
    postorder(ptr->rchild);
    printf("%d ",ptr->info);
}/*End of postorder()*/

```

**Output:**

```

1.Search
2.Insert
3.Delete
4.Preorder Traversal
5.Inorder Traversal
6.Postorder Traversal
7.Height of tree
8.Find minimum and maximum
9.Quit
Enter your choice: 5
20 30 50
1.Search
2.Insert
3.Delete
4.Preorder Traversal
5.Inorder Traversal
6.Postorder Traversal
7.Height of tree
8.Find minimum and maximum
9.Quit
Enter your choice: 6
50 30 20
1.Search
2.Insert
3.Delete
4.Preorder Traversal
5.Inorder Traversal
6.Postorder Traversal
7.Height of tree
8.Find minimum and maximum
9.Quit
Enter your choice: 7
Height of the tree is 3
1.Search
2.Insert
3.Delete
4.Preorder Traversal
5.Inorder Traversal
6.Postorder Traversal
7.Height of tree
8.Find minimum and maximum
9.Quit
Enter your choice: 8
Minimum key is 20
Maximum key is 50

```

```

1 /*Recursive operations in Binary Search Tree*/
2 #include<stdio.h>
3 #include<stdlib.h>
4 struct node
5 {
6     struct node *lchild;
7     int info;
8     struct node *rchild;
9 };
10 struct node *search(struct node *ptr,int skey);
11 struct node *insert(struct node *ptr,int ikey);
12 struct node *del(struct node *ptr,int dkey);
13 struct node *min(struct node *ptr);
14 struct node *max(struct node *ptr);
15 void preorder(struct node *ptr);
16 void inorder(struct node *ptr);
17 void postorder(struct node *ptr);
18 int height(struct node *ptr);
19
20 int main()
21 {
22     struct node *root=NULL,*ptr;
23     int choice,k;
24     while(1)
25     {
26         printf("\n\n");
27         printf("1.Search\n");
28         printf("2.Insert\n");
29         printf("3.Delete\n");
30         printf("4.Preorder Traversal\n");
31         printf("5.Inorder Traversal\n");
32         printf("6.Postorder Traversal\n");
33         printf("7.Height of tree\n");
34         printf("8.Find minimum and maximum\n");
35         printf("9.Quit\n");
36         printf("Enter your choice: ");
37         scanf("%d",&choice);
38         switch(choice)
39         {
40             case 1:
41                 printf("Enter the key to be searched: ");
42                 scanf("%d",&k);
43                 ptr = search(root,k);
44                 if(ptr==NULL)
45                     printf("Key not present\n");
46                 else
47                     printf("Key present\n");
48                 break;

```