Name : Sakshi. P. Khandoba
Sem : 3rd Section: 3C
Batch : 2
USN : 1BM19CS139

papergrid
Date: 21/12/20

LAB PROGRAM 12

Write a program
a) To construct a Binary Search Tree
b) To traverse the tree using all the methods i.e., in order, pre order and post order
c) To display the elements in the tree.

```
#include <stdio.h>
#include <process.h>
struct node
{
    int info;
    struct node *rlink;
    struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE) malloc(sizeof (struct node));
    if (x == NULL)
    {
        printf ("Memory is full. \n");
        exit(0);
    }
    return x;
}
void freenode (NODE x)
{
    free(x);
}
```

papergrid

Date: / /

```
NODE insert (NODE root, int item)
{
    NODE temp, cur, prev;
    temp = getnode();
    temp -> rlink = NULL;
    temp -> llink = NULL;
    temp -> info = item;
    if(root == NULL)
        return temp;
    prev = NULL;
    cur = root;
    while(cur != NULL)
    {
        prev = cur;
        cur = (item < cur -> info) ? cur->llink : cur->rlink;
    }
    if (item < prev -> info)
        prev -> llink = temp;
    else
        prev -> rlink = temp;
    return root;
}
NODE delete(NODE root, int item)
{
    NODE cur, parent, q, suc;
    if(root == NULL)
    {
        printf("Empty \n");
        return root;
    }
    parent = NULL;
    cur = root;
    while(cur != NULL && item != cur -> info)
    {
        parent = cur;
        cur = (item < cur->info) ? cur->llink: cur->rlink;
    }
    if(cur == NULL)
    {
        printf("Not found. \n");
        return root;
    }
    if(cur->llink == NULL)
        q = cur->rlink;
    else if(cur->~~ll~~rlink == NULL)
        q = cur->llink;
    else
    {
        suc = cur->rlink;
        while(suc->llink != NULL)
            suc = suc->llink;
        suc->llink = cur->llink;
        q = cur->rlink;
    }
    if(parent == NULL)
        return q;
    if(cur == parent->llink)
        parent->llink = q;
    else
        parent->rlink = q;
    freenode(cur);
    return root;
}
void preorder(NODE root)
{
    if(root != NULL)
```

```
        {
            printf("%d \n", root→info);
            preorder(root → llink);
            preorder (root → rlink):
        }
}
void postorder (NODE root)
{
        if(root != NULL)
        {
            postorder (root → llink);
            postorder (root → rlink);
            printf(" %d \n ", root → info);
        }
}
void inorder(NODE root)
{
        if(root != NULL)
        {
            inorder(root → llink);
            printf ("%d \n", root → info);
            inorder(root → rlink);
        }
}
void display (NODE root, int i)
{
        int j;
        if(root != NULL)
        {
            display(root → rlink, i+1);
            for(j=0; j<i; j++)
                printf(" ");
            printf ("%d \n", root → info);
        display(root → llink, i+1);
    }
}
void main()
{
    int item, choice;
    NODE root = NULL;
    for(;;)
    {
        printf("\n1. Insert \n2.Delete \n3. Preorder
        \n4.Postorder \n5. Inorder \n6. Display
        \n7. Exit \n");
        printf(" Enter the choice : ");
        scanf("%d ", &choice);
        switch(choice)
        {
        case 1: printf("Enter the item: ");
                scanf("%d", &item);
                root = insert (root, item);
                break;
        case 2: printf("Enter the item: ");
                scanf("%d", &item);
                root = delete ( root, item);
                break;
        case 3: printf("Preorder traversal: \n");
                preorder(root);
                break;
        case 4: printf("Postorder traversal : \n");
                postorder(root);
                break;
        case 5: printf(" Inorder traversal : \n");
                inorder(root);
                break;
```

papergrid

Date: / /

```
        case 6 : printf("Elements in the tree : \n");
                 display(root,0);
                 break;
        default: exit(0);
                 break;
        }
    }
}
```