

# PROGRAM 13

```
/* **** */
/* Name of the Program      : bst.c                               */
/* Aim                     : To implement binary search tree and perform its */
/*                         : operations                             */
/* Author                   : ASWIN BABU                          */
/* Date Written             : 28/03/2017                          */
/* Revision                 : 1                                    */
/* **** */
```

```
/* **** */
/* PROGRAM:                                                    */
/* data                    : To store the data about the tree     */
/* t                       : To store the traversed tree         */
/* right                   : To store the right node              */
/* left                    : To store the left node               */
/* root                    : To store the root node               */
/* d                       : To store node in which the operations has to be done */
/* **** */
```

```
#include<stdio.h>
#include<stdlib.h>
typedef struct BST
{
    int data;
    struct BST *left,*right;
}BST;
void inorder(BST *t)
{
    if(t!=NULL)
    {
        inorder(t->left);
        printf("%d ",t->data);
        inorder(t->right);
    }
}
int inordersucc(BST *t)
{
    BST *t1=t->right,*t2;int d;
    if(t1->left==NULL)
    {
        d=t1->data;
        t->right=NULL;
        free(t1);
        return d;
    }
}
```

```

}
while(t1->left!=NULL)
{
t2=t1;
t1=t1->left;
}
d=t1->data;
t2->left=NULL;
free(t1);
return d;
}
BST *getnode(int d)
{
BST *node=(BST*)malloc(sizeof(BST));
node->data=d;
node->left=node->right=NULL;
return node;
}
void insert(BST *t,int d)
{
if(d<t->data)
{
if(t->left==NULL)
t->left=getnode(d);
else
insert(t->left,d);
}
else if(d>t->data)
{
if(t->right==NULL)
t->right=getnode(d);
else
insert(t->right,d);
}
else
printf("Duplicate element:Cannot insert\n");
}
void search(BST *t,int d)
{
if(t==NULL)
{
printf("Element not found\n");
return;
}
if(d<t->data)
search(t->left,d);

```

```

else if(d>t->data)
    search(t->right,d);
else
    printf("Element found\n");
}
BST* delete(BST *t,int d)
{
    BST *cur=t,*par=t,*next_node,*new_node;int flag_root=0;
    while(cur!=NULL&&cur->data!=d)
    {
        if(d<cur->data)
        {
            par=cur;
            cur=cur->left;
        }
        else
        {
            par=cur;
            cur=cur->right;
        }
    }
    if(cur==NULL)
    {
        printf("Deleting element not found\n");
        return t;
    }
    if(cur==par)
        flag_root=1;
    if(cur->left==NULL&&cur->right==NULL)
    {
        if(par->left==cur)
            par->left=NULL;
        else if(par->right==cur)
            par->right=NULL;
        free(cur);
        t=NULL;
    }
    else if((cur->left==NULL&&cur->right!=NULL)||((cur->left!=NULL&&cur->right==NULL)))
    {
        if(cur->left!=NULL)
            new_node=cur->left;
        else
            new_node=cur->right;
        if(par->left==cur)
            par->left=new_node;
        else if(par->right==cur)

```

```

    par->right=new_node;
    if(flag_root==1)
        t=new_node;
    free(cur);
}
else
    cur->data=inordersucc(cur);
return t;
}
int main()
{
    BST *root=NULL;int d,c;
    while(1)
    {
        printf("Enter 1 to insert an element into BST\nEnter 2 to display\nEnter 3 to search\nEnter 4 to
delete\nEnter 5 to exit\nEnter your choice: ");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
                if(root==NULL)
                {
                    printf("Enter root element: ");
                    scanf("%d",&d);
                    root=getnode(d);
                }
            else
            {
                printf("Enter element: ");
                scanf("%d",&d);
                insert(root,d);
            }
            break;
            case 2:
                inorder(root);
                printf("\n");
            break;
            case 3:
                printf("\n");
                printf("Enter element to search: ");
                scanf("%d",&d);
                search(root,d);
            break;
            case 4:
                printf("Enter element to delete: ");
                scanf("%d",&d);

```



/\*\*\*\*\*\*  
/\*

**OUTPUT:**

Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 1  
Enter root element: 100  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 1  
Enter element: 90  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 1  
Enter element: 80  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 1  
Enter element: 50  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 2  
50 80 90 100  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 3

Enter element to search: 90

Element found  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 4  
Enter element to delete: 90  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 2  
50 80 100  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 0  
Invalid choice  
Enter 1 to insert an element into BST  
Enter 2 to display  
Enter 3 to search  
Enter 4 to delete  
Enter 5 to exit  
Enter your choice: 5 \*/

/\*  
\*\*\*\*\*  
\*/

# PROGRAM 15

```
/******  
/* Name of the Program      : merge_sort.c          */  
/* Aim                      : To implement merge sort */  
/* Author                   : ASWIN BABU           */  
/* Date Written             : 28/03/2017          */  
/* Revision                  : 1                   */  
/******
```

```
/******  
/* PROGRAM:                                     */  
/* a[]                      : To store the array elements */  
/* n                         : To store the no of elements in the array */  
/* i                         : For loop iteration          */  
/* i1,j1                     : Temporary variables         */
```

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

```
void mergesort(int a[],int i,int j);  
void merge(int a[],int i1,int j1,int i2,int j2);
```

```
int main()  
{  
    int a[30],n,i;  
    printf("Enter no of elements :");  
    scanf("%d",&n);  
    printf("Enter array elements:\n");
```

```
    for(i=0;i<n;i++)  
        scanf("%d",&a[i]);
```

```
    mergesort(a,0,n-1);
```

```
    printf("\nSorted array is :");  
    for(i=0;i<n;i++)  
        printf("%d ",a[i]);
```

```
    return 0;  
}
```

```
void mergesort(int a[],int i,int j)  
{  
    int mid;  
  
    if(i<j)
```

```
{
    mid=(i+j)/2;
    mergesort(a,i,mid);
    mergesort(a,mid+1,j);
    merge(a,i,mid,mid+1,j);
}
```

```
void merge(int a[],int i1,int j1,int i2,int j2)
```

```
{
    int temp[50];
    int i,j,k;
    i=i1;
    j=i2;
    k=0;

    while(i<=j1 && j<=j2)
    {
        if(a[i]<a[j])
            temp[k++]=a[i++];
        else
            temp[k++]=a[j++];
    }

    while(i<=j1)
        temp[k++]=a[i++];

    while(j<=j2)
        temp[k++]=a[j++];

    for(i=i1,j=0;i<=j2;i++,j++)
        a[i]=temp[j];
}
```

```
/****/
```

```
/******  
/*
```

```
/*
```

**OUTPUT:**

Enter no. of elements :5

Enter array elements :

10

90

50

30

80

Sorted array is: 10 30 50 80 90 \*/

```
/******  
/*
```