Disha.N
1BM19CS051

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

Struct node
{
    int data;
    Struct node *left ;
    Struct node *right ;
};
typedef struct node *NODE;
NODE getnode(int data)
{
    NODE x = (NODE)malloc(sizeof(struct node));
    x->data= data;
    x->right= NULL;
    x->left= NULL;
    return x;
}
NODE insert(NODE root, int info)
{
    if(root==NULL)
    {
        root =getnode(info);
    }   return root;
    else if (info<= root->data)
    {
        root->left = insert(root->left, info);
    }
    else
    {
        root->right= insert(root->right, info);
    return root;
}

void preorder(NODE root)
{
    if(root ==NULL)
        return;
    printf("%d\t", root->data);
    preorder (root->left);
    preorder( root-> right);
}
```

```
void inorder(NODE root)
  {    if(root=NULL)
           return;
       inorder(root->left);
       printf("%d\t", root ->data);
       inorder(root->right);
    }
   void postorder( NODE root)
  {
       if(root==NULL)
           return;
        postorder(root->left);
        postorder( root->right);
        printf("%d\t", root->data);
     }

 NODE findmin(NODE root)
 {
     if(root == NULL)
      { return NULL;
      }
     else if (root->left == NULL)
        { return root;
        }
     ~~else if (root~~
         return findmin(root->left);
 }

NODE delete_node(NODE root, int info)
 {
     if(root ==NULL)
         return root;
     else if (info< root->data)
       {
         root->left =delete-node(root->left, info);
       }
      else if (info> root->data)
       {
         root->right = delete=node(root->right, info);
       }
     else{
          if(root->left == NULL &&root-> right== NULL)
    {
        free(root);
         root = NULL;
          return root;
       }
   else if (root->left ==NULL)
       {
          NODE temp = root;
           root = root->left;
             free(temp);
              return root;
         }
     else if (root-> right = = NULL)
        {   NODE temp= root;
             root= root->right;
               free(temp);
              } return root;
     else{ ~~if (root ->~~
        NODE temp = findmin(root->right);
           root->data= temp->data;
         root->right= delete_node(root->right,temp->data)
              return root;
           }
    }
}

  Void display(NODE root , int i)
{
    if(root== NULL)
         return;
    display(root->right, i+1);
    for(int j=1 ; j<=i; j++)
            printf(" ");
       printf("%d\n", root->data);
          display(root->left, i+1);
     }
int main()
{    NODE root = NULL;
     int data , option;
   do{
         printf(" ");
       }
   return 0;
}
```