DSA LAB DA2

REG NO: 18BCE0555

NAME :HIMANSHU RUWATIA

SLOT: G2

BINARY TREE :-

```
#include  <iostream>
using namespace std;
struct node
{
   int x;struct node *l=NULL,*r=NULL;
};
struct node* searching(struct node *root,int a)
{
   if(root==NULL)
      return NULL;
   if(root->x==a)
      return root;
   struct node *x=searching(root-
   >l,a); if(x!=NULL)return x;
   return searching(root->r,a);
}
struct node* insertion(struct node *root)
{
   int a;cin>>a;
   if(root==NULL)
   {
      root=new struct node;
      root->x=a;return root;
   }
   cout<<"enter the element after which you want to insert\n";
   int b;cin>>b;
   struct node
   *t=searching(root,b);
   if(t==NULL)
   {
      cout<<"element not present\n";return root;
   }
   cout<<"enter 1 for left and 2 for right\n";
   cin>>b;
   if(b==1)
   {
      if(t->l!=NULL)
      {
          cout<<"position already occupied\n";return root;
      }
      t->l=new struct node;
      t->l->x=a;
   }
   else if(b==2)
   {
      if(t->r!=NULL)
      {
          cout<<"position already occupied\n";return root;
      }
      t->r=new struct node;
      t->r->x=a;
   }
   return root;
}
void inorder(struct node *root)
{
   if(root==NULL)
      return;
   inorder(root->l);
   cout<<root->x<<"\t";
   inorder(root->r);
}
void postorder(struct node *root)
{
   if(root==NULL)
      return;
   postorder(root->l);
   postorder(root->r);
   cout<<root->x<<"\t";
}
void preorder(struct node *root)
{
   if(root==NULL)
      return;
   cout<<root->x<<"\t";
   preorder(root->l);
   preorder(root->r);
}
int main()
{
   struct node *root=NULL;
   int c=0;
   while(c!=4)
   {
      cout<<"1-enter new element\n2-search for element\n3-display\n4-exit\n";
      cin>>c;
      switch(c)
      {
          case 1:
              root=insertion(r
              oot); break;
          case 2:
              if(root==NULL)
              {
                 cout<<"tree empty\n";continue;
              }
              int a;cin>>a;
              if(searching(root,a)==NULL)cout<<"not
              found\n"; else cout<<"found\n";
              break;
          case 3:
              if(root==NULL)
              {
                 cout<<"tree empty\n";continue;
              }
              cout<<"1-inorder\n2-preorder\n3-postorder\n";
              cin>>c;
              if (c==1)
                   inorder(root);
              else if(c==2)
                   preorder(root);
               else
                   postorder(root);
              cout<<endl;
              break;
          case 4:
              return 0;
          default:
              cout<<"Wrong choice\n";
      }
   }
}
```

Terminal:-

```
1-enter new element
2-search for element
3-display
4-exit
1
4
1-enter new element
2-search for element
3-display
4-exit
1
0
enter the element after which you want to insert
4
enter 1 for left and 2 for right
1
1-enter new element
2-search for element
3-display
4-exit
1
3
enter the element after which you want to insert
```

```
0
enter 1 for left and 2 for right
2
1-enter new element
2-search for element
3-display
4-exit
1
65
enter the element after which you want to insert
4
enter 1 for left and 2 for right
2
1-enter new element
2-search for element
3-display
4-exit
2
3
found
1-enter new element
2-search for element
3-display
4-exit
3
1-inorder
2-preorder
3-postorder
1
0       3       4       65
1-enter new element
2-search for element
3-display
4-exit
3
1-inorder
2-preorder
3-postorder
2
4       0       3       65
1-enter new element
2-search for element
3-display
4-exit
3
1-inorder
2-preorder
3-postorder
3
3       0       65      4
1-enter new element
2-search for element
3-display
4-exit
4
```

```
------------------
(program exited with code: 0)
Press return to continue
```

BINARY SEARCH TREE :-

```
#include <iostream>
using namespace std;
struct node
{
int x;struct node *l=NULL,*r=NULL;
};
struct node* searching(struct node *root, int a)
{
   if(root==NULL)
      return NULL;
   if(root->x==a)
      return root;
   else if(root->x>a)
      return searching(root->l,a);
   else
      return searching(root->r,a);
}
struct node* insertion(struct node *root,int a)
{
   if(root==NULL)
   {
      root=new struct node;root->x=a;return root;
   }
   if(a>root->x)
      root->r=insertion(root->r,a);
   else
      root->l=insertion(root->l,a);
   return root;
}
struct node* inorder_next(struct node *root)
{

   if(root==NULL)
      return NULL;
   while(root->l!=NULL)
      root=root->l;
   return root;
}
struct node* del(struct node *root,int a)
{
   struct node *t;
   if(root==NULL)
      cout<<"element not found";
   else if(root->x>a)
      root->l=del(root->l,a);
   else if(root->x<a)
      root->r=del(root->r,a);
   else if(root->x==a)
```

```
    {
        if(root->l!=NULL&&root->r!=NULL)
        {
            t=inorder_next(root->r);
            root->x=t->x;
            root->r=del(root->r,root->x);
        }
        else
        {
            t=root;
            if(root->l!=NULL)
                root=root->l;
            else if(root->r!=NULL)
                root=root->r;
            else
                root=NULL;
            free(t);
        }
    }
    return root;
}
void inorder(struct node *root)
{
    if(root==NULL)
        return;
    inorder(root->l);
    cout<<root->x<<"\t";
    inorder(root->r);
}
void postorder(struct node *root)
{
    if(root==NULL)
        return;
    postorder(root->l);
    postorder(root->r);
    cout<<root->x<<"\t";
}
void preorder(struct node *root)
{
    if(root==NULL)
        return;
    cout<<root->x<<"\t";
    preorder(root->l);
    preorder(root->r);
}
int main()
{
    struct node *root=NULL; int c=0;
    while(c!=5)
    {
        cout<<"1-enter new element\n2-search for element\n3-display\n4-delete\n5-exit\n";
        cin>>c;
        switch(c)
        {
            case 1:
                int ab;cin>>ab;
                root=insertion(root,ab);
                break;
            case 2:
                if(root==NULL)
                {
                    cout<<"tree empty\n";
                    continue;
                }
                int a;
                cin>>a;
                if(searching(root,a)==NULL)
                    cout<<"not found\n";
                else
                    cout<<"found\n";
                break;
            case 3:
                if(root==NULL)
                {
                    cout<<"tree empty\n";
                    continue;
                }
                cout<<"1-inorder\n2-preorder\n3-postorder\n";
                cin>>c;
                if (c==1)
                      inorder(root);
                else if(c==2)
                      preorder(root);
                else
                      postorder(root);
                cout<<endl;
                break;
            case 4:
                if(root==NULL)
                {
                    cout<<"tree empty\n";
                    continue;
                }
                cin>>a;
                root=del(root,a);
                break;
            case 5:
                return 0;
            default:
                cout<<"Wrong choice\n";
        }
    }
}
```

Terminal:-

```
1-enter new element
2-search for element
3-display
4-delete
5-exit
1
```

```
6
1-enter new element
2-search for element
3-display
4-delete
5-exit
1
3
1-enter new element
2-search for element
3-display
4-delete
5-exit
1
8
1-enter new element
2-search for element
3-display
4-delete
5-exit
1
4
1-enter new element
2-search for element
3-display
4-delete
5-exit
1
9
1-enter new element
2-search for element
3-display
4-delete
5-exit
3
1-inorder
2-preorder
3-postorder
1
3       4       6       8       9
1-enter new element
2-search for element
3-display
4-delete
5-exit
3
1-inorder
2-preorder
3-postorder
2
6       3       4       8       9
1-enter new element
2-search for element
3-display
4-delete
5-exit
```

```
3
1-inorder
2-preorder
3-postorder
3
4       3       9       8       6
1-enter new element
2-search for element
3-display
4-delete
5-exit
4
8
1-enter new element
2-search for element
3-display
4-delete
5-exit
3
1-inorder
2-preorder
3-postorder
1
3       4       6       9
1-enter new element
2-search for element
3-display
4-delete
5-exit
3
1-inorder
2-preorder
3-postorder
2
6       3       4       9
1-enter new element
2-search for element
3-display
4-delete
5-exit
3
1-inorder
2-preorder
3-postorder
3
4       3       9       6
1-enter new element
2-search for element
3-display
4-delete
5-exit
```