**NAME :** JAGADEESH R
**REGNO :** 2021506314
**ADS LAB08 :** SPLAY TREE-Insertion, Deletion, Find, Display

**SOURCE CODE :**

```
#include <iostream>
using namespace std;
struct s
{
  int k;
  s* lch;
  s* rch;
};
class SplayTree
{
  public:
  s* RR_Rotate(s* k2)
  {
    s* k1 = k2->lch;
    k2->lch = k1->rch;
    k1->rch = k2;
    return k1;
  }
  s* LL_Rotate(s* k2)
  {
    s* k1 = k2->rch;
    k2->rch = k1->lch;
    k1->lch = k2;
    return k1;
  }
  s* Splay(int key, s* root)
  {
    if (!root)
    return NULL;
    s header;
    header.lch= header.rch = NULL;
    s* LeftTreeMax = &header;
    s* RightTreeMin = &header;
    while (1)
    {
      if (key < root->k)
      {
        if (!root->lch)
        break;
        if (key< root->lch->k)
        {
```

```
            root = RR_Rotate(root);
            if (!root->lch)
            break;
          }
          RightTreeMin->lch= root;
          RightTreeMin = RightTreeMin->lch;
          root = root->lch;
          RightTreeMin->lch = NULL;
        }
        else if (key> root->k)
        {
          if (!root->rch)
          break;
          if (key > root->rch->k)
          {
            root = LL_Rotate(root);
            if (!root->rch)
            break;
          }
          LeftTreeMax->rch= root;
          LeftTreeMax = LeftTreeMax->rch;
          root = root->rch;
          LeftTreeMax->rch = NULL;
        }
        else
        break;
      }
     //   LeftTreeMax→rch = root->lch;
     //   RightTreeMin→lch = root->rch;
      root->lch = header.rch;
      root->rch = header.lch;
      return root;
    }
    s* New_Node(int key)
    {
      s* p_node = new s;
      if (!p_node)
      {
        fprintf(stderr, "Out of memory!\n");
        exit(1);
      }
      p_node->k = key;
      p_node->lch = p_node->rch = NULL;
      return p_node;
    }
    s* Insert(int key, s* root)
    {
```

```
  static s* p_node = NULL;
  if (!p_node)
  p_node = New_Node(key);
  else
  p_node->k = key;
  if (!root)
  {
    root = p_node;
    p_node = NULL;
    return root;
  }
  root = Splay(key, root);
  if (key < root->k)
  {
    p_node->lch= root->lch;
    p_node->rch = root;
    root->lch = NULL;
    root = p_node;
  }
  else if (key > root->k)
  {
    p_node->rch = root->rch;
    p_node->lch = root;
    root->rch = NULL;
    root = p_node;
  }
  else
  return root;
  p_node = NULL;
  return root;
}
s* Delete(int key, s* root)//delete node
{
  s* temp;
  if (!root)//if tree is empty
  return NULL;
  root = Splay(key, root);
  if (key != root->k)//if tree has one item
  return root;
  else
  {
    if (!root->lch)
    {
      temp = root;
      root = root->rch;
    }
    else
```

```
        {
          temp = root;
          root = Splay(key, root->lch);
          root->rch = temp->rch;
        }
        free(temp);
        return root;
      }
    }
    s* Search(int key, s* root)//seraching
    {
      return Splay(key, root);
    }
    void InOrder(s* root)
    {
      if (root)
      {
        InOrder(root->lch);
        cout<< "key: " <<root->k;
        if(root->lch)
        cout<< " | left child: "<< root->lch->k;
        if(root->rch)
        cout << " | right child: " << root->rch->k;
        cout<< "\n";
        InOrder(root->rch);
      }
    }
};
int main()
{
    SplayTree st;
    s *root;
    root = NULL;
    st.InOrder(root);
    int i, c;
    while(1)
    {
      cout<<"\n"<<"1. Insert an element into the tree"<<endl;
      cout<<"2. Delete an element in the tree"<<endl;
      cout<<"3. Find an element in the tree"<<endl;
      cout<<"4. Exit"<<endl;
      cout<<"Enter your choice : ";
      cin>>c;
      switch(c)
      {
        case 1:
          cout<<"Enter an element to be inserted : ";
```

```
            cin>>i;
            root = st.Insert(i, root);
            cout<<"\nElement Inserted : "<<i<<endl;
            st.InOrder(root);
            break;
          case 2:
            cout<<"Enter an element for deletion : ";
            cin>>i;
            root = st.Delete(i, root);
            cout<<"\nElement Deleted : "<<i<<endl;
            st.InOrder(root);
            break;
          case 3:
            cout<<"Enter an element to perform search : ";
            cin>>i;
            root = st.Search(i, root);
            cout<<"\nElement Searched : "<<i<<endl;
            st.InOrder(root);
            break;
          case 4:
          cout<<"Inorder : "<<endl;
            st.InOrder(root);
            cout<<"Exiting code..."<<endl;
            exit(0);
            break;
          default:
            cout<<"\nInvalid option : "<<c<<endl;
        }
      }
      return 0;
    }
```

**OUTPUT :**

```
Output
/tmp/57fX6OXEHK.o
1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 1
Enter an element to be inserted : 2
Element Inserted : 2
key: 2

1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 1
Enter an element to be inserted : 5
Element Inserted : 5
key: 2
key: 5 | left child: 2

1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 1
Enter an element to be inserted : 9
Element Inserted : 9
key: 5
key: 9 | left child: 5
```

```
1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 1
Enter an element to be inserted : 7
Element Inserted : 7
key: 5
key: 7 | left child: 5 | right child: 9
key: 9

1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 1
Enter an element to be inserted : 4
Element Inserted : 4
key: 4 | right child: 5
key: 5

1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 2
Enter an element for deletion : 4
Element Deleted : 4

1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 3
Enter an element to perform search : 9
Element Searched : 9

1. Insert an element into the tree
2. Delete an element in the tree
3. Find an element in the tree
4. Exit
Enter your choice : 4
```