NLDS
Lab 8

Akanksha Agarwal
19103242
B8

```
#include <bits/stdc++.h>
using namespace std;

class node {
    public:
    int key;
    node *left, *right;
};

node* newNode(int key) {
    node* Node = new node();
    Node->key = key;
    Node->left = Node->right = NULL;
    return Node;
}

node *rightRotate(node *x) {
    node *y = x->left;
    x->left = y->right;
    y->right = x;
    return y;
}

node *leftRotate(node *x) {
    node *y = x->right;
    x->right = y->left;
    y->left = x;
    return y;
}

node *splay(node *root, int key) {

    if (root == NULL || root->key == key)
        return root;
```

```
if (root->key > key){

    if (root->left == NULL)
    return root;

    if (root->left->key > key) {

        root->left->left = splay(root->left->left, key);
        root = rightRotate(root);
    }

    else if (root->left->key < key) {
        root->left->right = splay(root->left->right, key);

        if (root->left->right != NULL)
            root->left = leftRotate(root->left);
    }

    return (root->left == NULL)? root: rightRotate(root);
}
else {

    if (root->right == NULL)
    return root;

    if (root->right->key > key){

        root->right->left = splay(root->right->left, key);

        if (root->right->left != NULL)
            root->right = rightRotate(root->right);
    }
    else if (root->right->key < key) {
        root->right->right = splay(root->right->right, key);
        root = leftRotate(root);
    }

    return (root->right == NULL)? root: leftRotate(root);
}
```

```
}


node *insert(node *root, int k) {

   if (root == NULL)
   return newNode(k);

   root = splay(root, k);

   if (root->key == k)
   return root;

   node *newnode = newNode(k);

   if (root->key > k) {
      newnode->right = root;
      newnode->left = root->left;
      root->left = NULL;
   }

   else {
      newnode->left = root;
      newnode->right = root->right;
      root->right = NULL;
   }

   return newnode;
}

void preOrder(node *root) {
   if (root != NULL) {
      cout<<root->key<<" ";
      preOrder(root->left);
      preOrder(root->right);
   }
}

node *search(node *root, int key) {
   return splay(root, key);
```

```
}

struct node* delete_key(struct node *root, int key){
    struct node *temp;
    if (!root)
        return NULL;

    root = splay(root, key);

    if (key != root->key)
        return root;

    if (!root->left){
        temp = root;
        root = root->right;
    }

    else{
        temp = root;

        root = splay(root->left, key);
        root->right = temp->right;
    }

    free(temp);
    return root;
}

int main() {
    node *root = NULL;
    root = insert(root, 20);
    root = insert(root, 15);
    root = insert(root, 32);
    root = insert(root, 37);
    root = insert(root, 14);
    root = insert(root, 8);
    root = insert(root, 6);
    root = insert(root, 23);
    root = insert(root, 29);
    root = insert(root, 12);
    cout<<"Preorder traversal of the modified Splay tree is \n";
    preOrder(root);
    cout << endl << endl;

    root = search(root, 23);
    cout<<"Preorder traversal of the modified Splay tree is \n";
    preOrder(root);
    cout << endl << endl;

    root = delete_key(root, 8);
    cout<<"Preorder traversal of the modified Splay tree is \n";
    preOrder(root);
    cout << endl << endl;

    return 0;
}
```