EAST WEST UNIVERSITY

Summer 2021 Semester

Course Code: CSE207

Course Title: Data Structure

Project

Splay tree

Submitted To

Tanni Mittra

Senior Lecturer

Department of Computer Science and Engineering

Submitted By

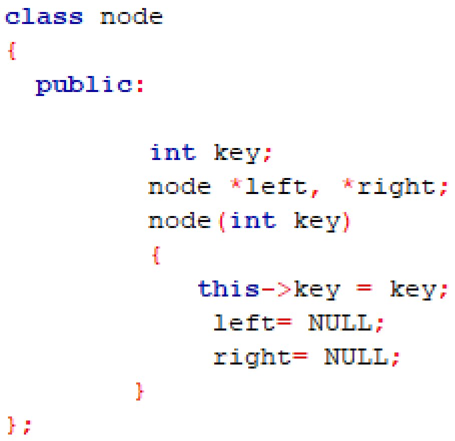
Group: 2

K. M. Safin Kamal **(2020-1-60-235)**

Ishmam-ur-Rahman **(2020-1-60-241)**

Al Hossain **(2020-1-60-016)**

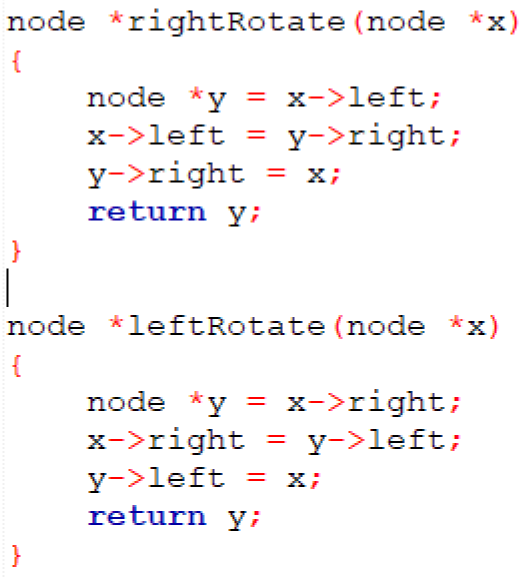
-Node header file. Here every node has a key(data) and left-right node.



```
class node
{
  public:

          int key;
          node *left, *right;
          node(int key)
          {
              this->key = key;
               left= NULL;
               right= NULL;
         }
};
```

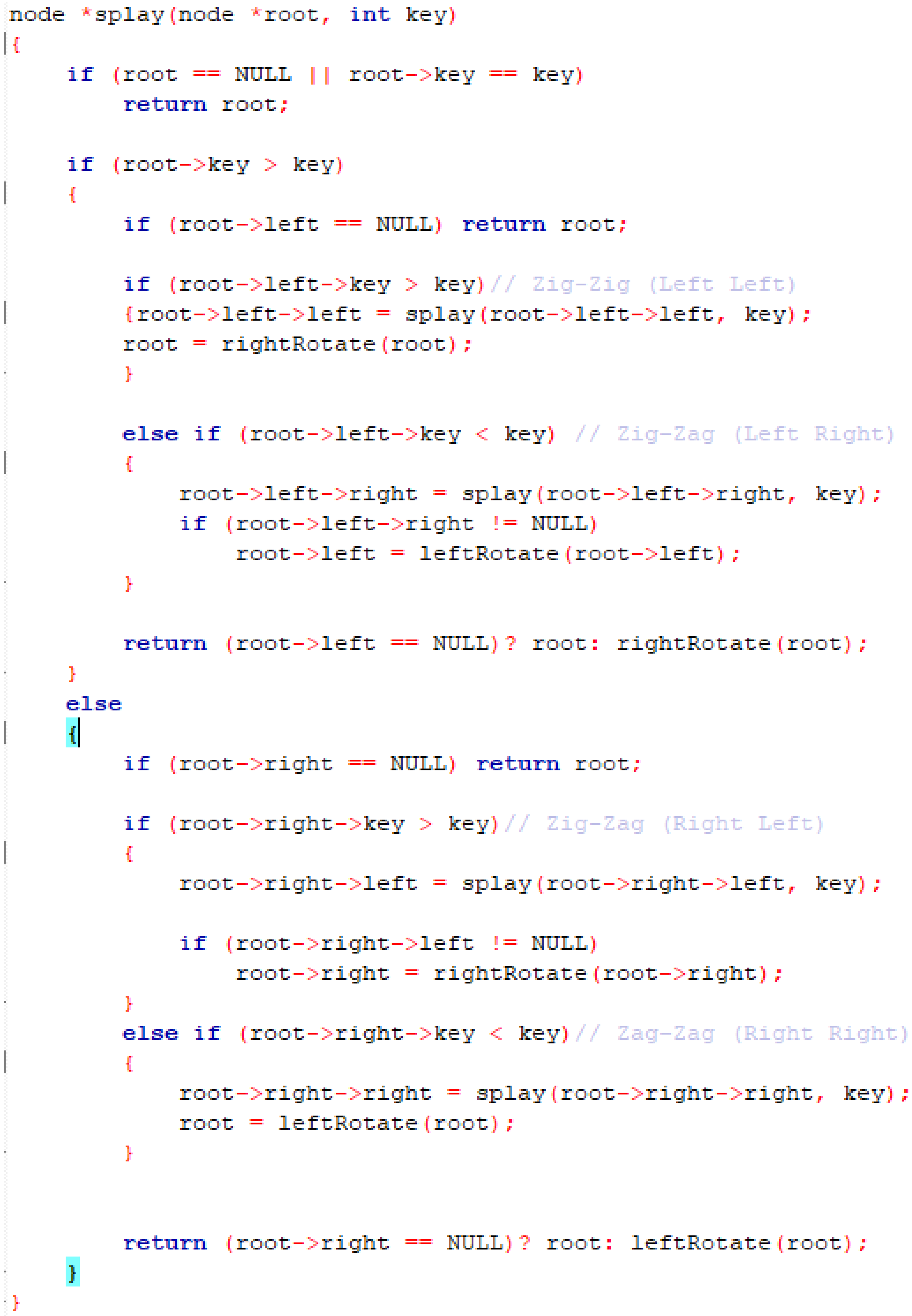
-Then we create the important header file Splay.h . In this header file node.h is included.

In splay header file we have created some functions. All functions are in SPLAY class. First two are for rotation (left and right).



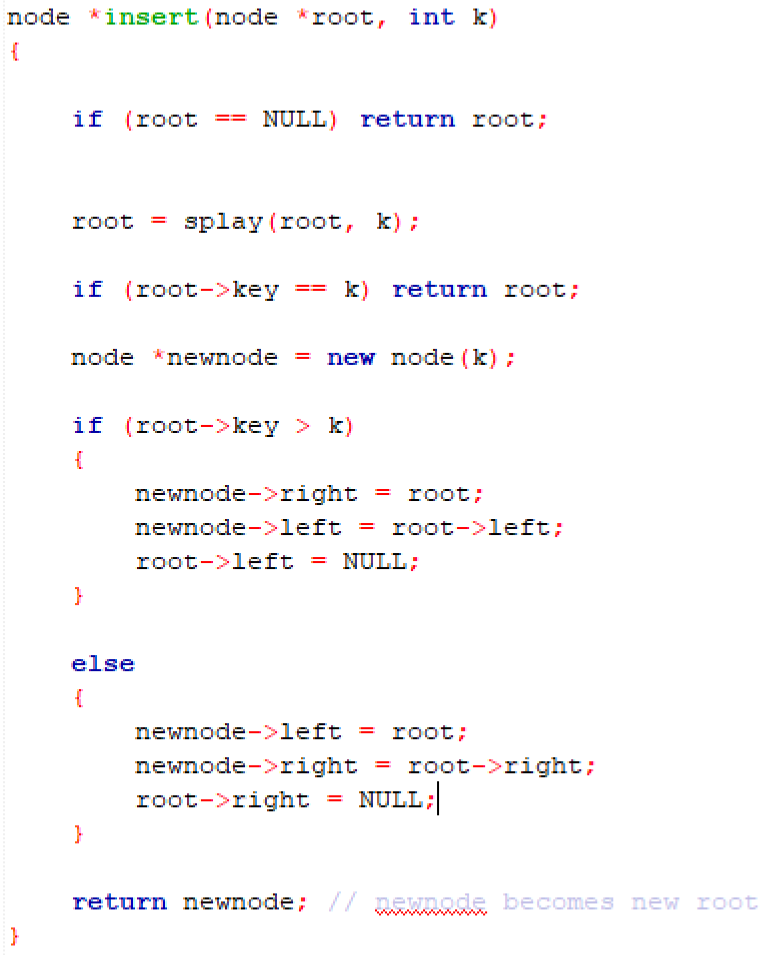
```
node *rightRotate(node *x)
{
    node *y = x->left;
    x->left = y->right;
    y->right = x;
    return y;
}

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

-Then we implement the splay function. This function brings the key at root if key is present in tree. If key is not present, then it brings the last accessed item atroot. This function modifies the tree and returns the new root.



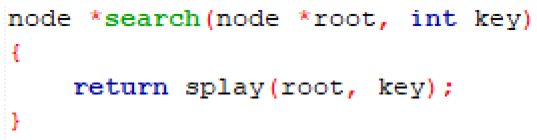
```
node *splay(node *root, int key)
{
    if (root == NULL || root->key == key)
        return root;

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

        if (root->left->key > key)// Zig-Zig (Left Left)
        {root->left->left = splay(root->left->left, key);
        root = rightRotate(root);
        }

        else if (root->left->key < key) // Zig-Zag (Left Right)
        {
            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)// Zig-Zag (Right Left)
        {
            root->right->left = splay(root->right->left, key);

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


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

-There are many operation in splay tree. One of them is insert. Here we created the insert function. Function to insert a new key “ k ” in splay tree with given root.



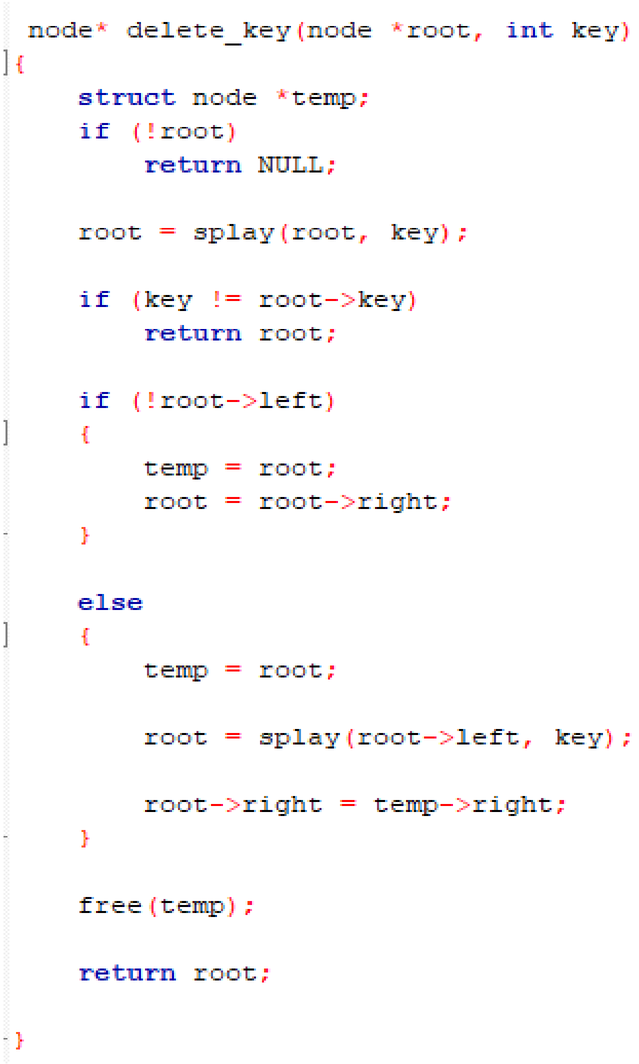
```
node *insert(node *root, int k)
{

    if (root == NULL) return root;


    root = splay(root, k);

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

    node *newnode = new node(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; // newnode becomes new root
}
```

-Our next function is search. Here the key will be searched and will be root by splaying



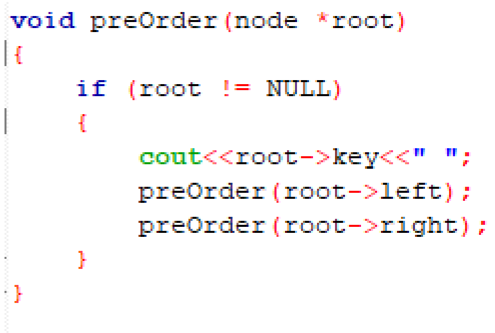
```
node *search(node *root, int key)
{
    return splay(root, key);
}
```

-Our last function for splay tree operation is delete



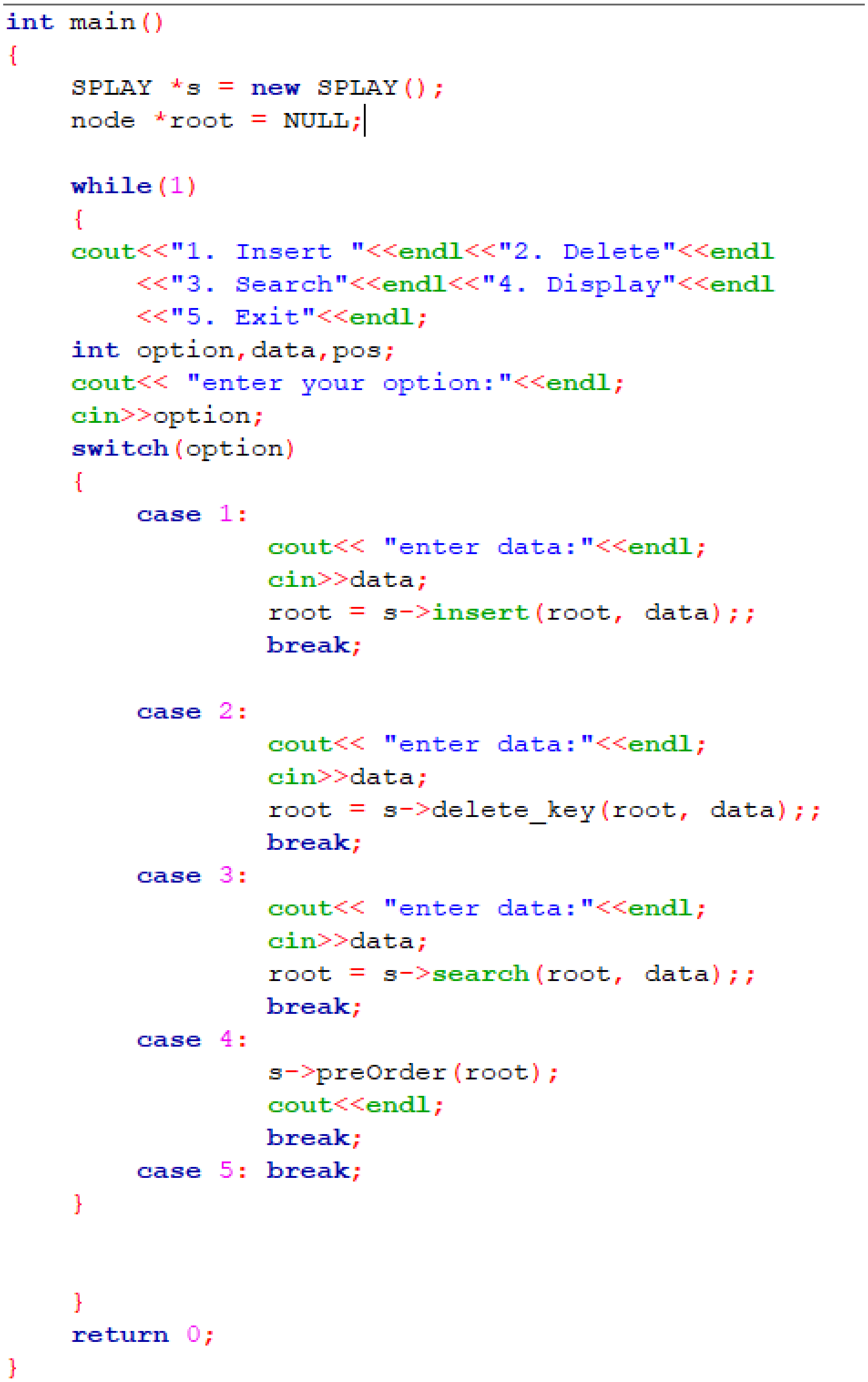
```
node* delete_key(node *root, int key)
{
    struct node *temp;
    if (!root)
        return NULL;

    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;

}
```

-To display the tree we used pre-order method.



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

-The main function:::



```
int main()
{
    SPLAY *s = new SPLAY();
    node *root = NULL;

    while(1)
    {
    cout<<"1. Insert "<<endl<<"2. Delete"<<endl
        <<"3. Search"<<endl<<"4. Display"<<endl
        <<"5. Exit"<<endl;
    int option,data,pos;
    cout<< "enter your option:"<<endl;
    cin>>option;
    switch(option)
    {
        case 1:
                cout<< "enter data:"<<endl;
                cin>>data;
                root = s->insert(root, data);;
                break;

        case 2:
                cout<< "enter data:"<<endl;
                cin>>data;
                root = s->delete_key(root, data);;
                break;
        case 3:
                cout<< "enter data:"<<endl;
                cin>>data;
                root = s->search(root, data);;
                break;
        case 4:
                s->preOrder(root);
                cout<<endl;
                break;
        case 5: break;
    }


    }
    return 0;
}
```