SARVESH BAPUSAHEB CHAVAN

ROLL NO:SYCOA124

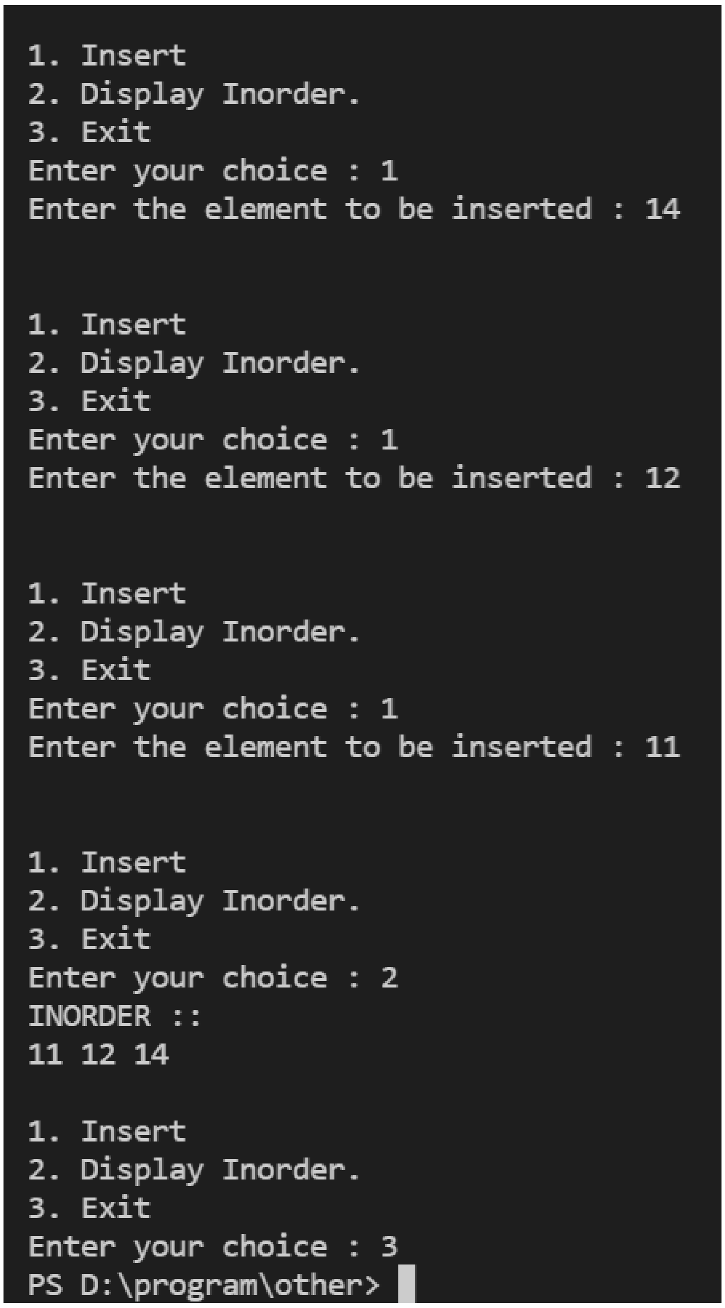
CODE:

```
#include <iostream>
#include <climits>

using namespace std;

struct node
{
    struct node *left;
    struct node *right;
    int data;
} * root;

class AVL_Tree
{
public:
    node *root;
    AVL_Tree()
    {
        root = NULL;
    }

    node *balance(node *);
    node *insert(node *, int);
    void inorder(node *);
    void preorder(node *);
    int balance_factor(node *);
    int height(node *);
    node *LL_rotation(node *);
    node *RR_rotation(node *);
    node *LR_rotation(node *);
    node *RL_rotation(node *);
    int difference(node *);
};

int AVL_Tree::height(node *temp)
{
    int h = 0;
    if (temp != NULL)
    {
        int max_height = max(height(temp->left), height(temp->right));
        h = max_height + 1;
    }
```

```cpp
    return h;
}

int AVL_Tree::difference(node *temp)
{
    int bf = height(temp->left) - height(temp->right);
    return bf;
}

node *AVL_Tree::LL_rotation(node *parent)
{
    node *temp;
    temp = parent->left;
    parent->left = temp->right;
    temp->right = parent;

    return temp;
}

node *AVL_Tree::RR_rotation(node *parent)
{
    node *temp;
    temp = parent->right;
    parent->right = temp->left;
    temp->left = parent;

    return temp;
}

node *AVL_Tree::LR_rotation(node *parent)
{
    node *temp;
    parent->left = RR_rotation(parent->right);
    parent = LL_rotation(parent);

    return parent;
}

node *AVL_Tree::RL_rotation(node *parent)
{
    node *temp;
    parent->right = LL_rotation(parent->left);
    parent = RR_rotation(parent);

    return parent;
}

void AVL_Tree::inorder(node *temp)
```

```cpp
{
    if (temp != NULL)
    {
        inorder(temp->left);
        cout << temp->data << " ";
        inorder(temp->right);
    }
}


node *AVL_Tree::balance(node *temp)
{
    int bf = difference(temp);
    if (bf > 1)
    {
        if (difference(temp->left) > 0)
            temp = LL_rotation(temp);
        else
            temp = LR_rotation(temp);
    }
    else if (bf < -1)
    {
        if (difference(temp->right) > 0)
            temp = RL_rotation(temp);
        else
            temp = RR_rotation(temp);
    }
    return temp;
}

node *AVL_Tree::insert(node *nn, int value)
{
    if (nn == NULL)
    {
        nn = new node;
        nn->data = value;
        nn->left = NULL;
        nn->right = NULL;

        return nn;
    }
    else if (value < nn->data)
    {
        nn->left = insert(nn->left, value);
        nn = balance(nn);
    }
    else if (value >= nn->data)
    {
        nn->right = insert(nn->right, value);
        nn = balance(nn);
    }
    return nn;
}
int main()
{
    int choice, num;
    AVL_Tree a;

    while (choice != 4)
    {
        cout << endl
             << endl;
        cout << "1. Insert " << endl;
        cout << "2. Display Inorder." << endl;
        cout << "3. Exit";

        cout << endl;
        cout << "Enter your choice : ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            int num, n;
            cout << "Enter the element to be inserted : ";
            cin >> num;
            root = a.insert(root, num);
            break;

        case 2:
            cout << "INORDER ::" << endl;
            a.inorder(root);
            break;

        case 3:
            return 0;
            break;

        default:
            cout << "INVALID INPUT!" << endl;
        }
    }

    return 0;
}
```

OUTPUT:



```
1. Insert
2. Display Inorder.
3. Exit
Enter your choice : 1
Enter the element to be inserted : 14


1. Insert
2. Display Inorder.
3. Exit
Enter your choice : 1
Enter the element to be inserted : 12


1. Insert
2. Display Inorder.
3. Exit
Enter your choice : 1
Enter the element to be inserted : 11


1. Insert
2. Display Inorder.
3. Exit
Enter your choice : 2
INORDER ::
11 12 14

1. Insert
2. Display Inorder.
3. Exit
Enter your choice : 3
PS D:\program\other>
```