1. **Lower Bound-STL**

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

*/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/*

    int n;

    cin >> n;

    vector<int> v(n);

    for (int i = 0; i < n; i++) {

        cin >> v[i];

    }

    int q;

    cin >> q;

    for (int i = 0; i < q; i++) {

        int x;

        cin >> x;

        vector<int>::iterator low = lower\_bound(v.begin(), v.end(), x);

        if (v[low - v.begin()] == x) {

            cout << "Yes " << (low - v.begin() + 1) << endl;

        } else {

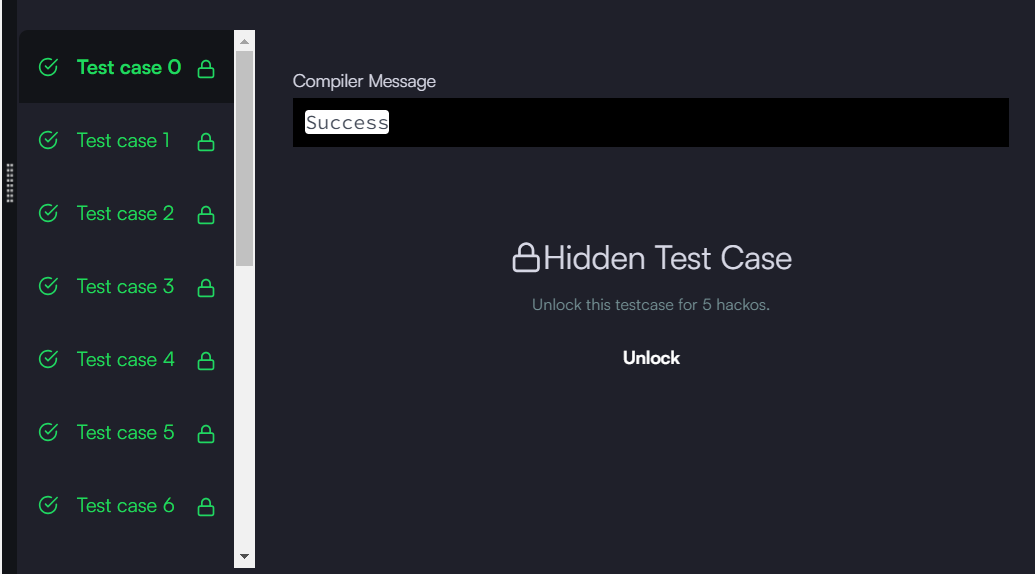
            cout << "No " << (low - v.begin() + 1) << endl;

        }

    }

    return 0;

}



1. **Sets-STL**

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <set>

#include <algorithm>

using namespace std;

int main() {

*/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/*

    set<int> s;

    int Q;

    cin>>Q;

    for(int i=0; i<Q; i++){

        int query;

        int x;

        cin>>query;

        cin>>x;

        switch (query) {

        case 1:

            s.insert(x);

            break;

        case 2:

            s.erase(x);

            break;

        case 3:

            auto itr=s.find(x);

            if(itr !=s.end()) {

                cout<<"Yes"<<endl;

            }

            else{

                 cout<<"No"<<endl;

            }

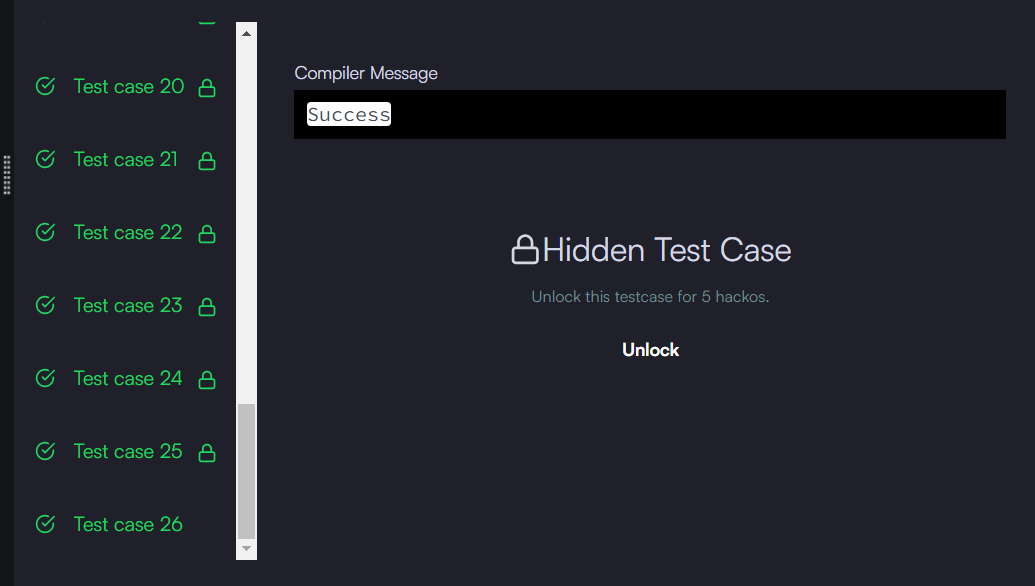
            break;

        }

    }

    return 0;

}



1. **Maps-STL**

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <set>

#include <map>

#include <algorithm>

using namespace std;

int main() {

*/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/*

   int n;

    int query;

    map<string, int> students;

    cin>>n;

    string name;

    int marks;

    for(int i=0;i<n;i++)

    {

        cin>>query;

        if(query==1)

        {

            cin>>name>>marks;

            students[name]+=marks;

        }

        if(query==2 || query==3)

        {

            cin>>name;

        }

        switch(query)

        {

        case 1:

        students.insert(make\_pair(name,marks));

        break;

        case 2:

        students.erase(name);

        break;

        case 3:

        cout<<students[name]<<"\n";

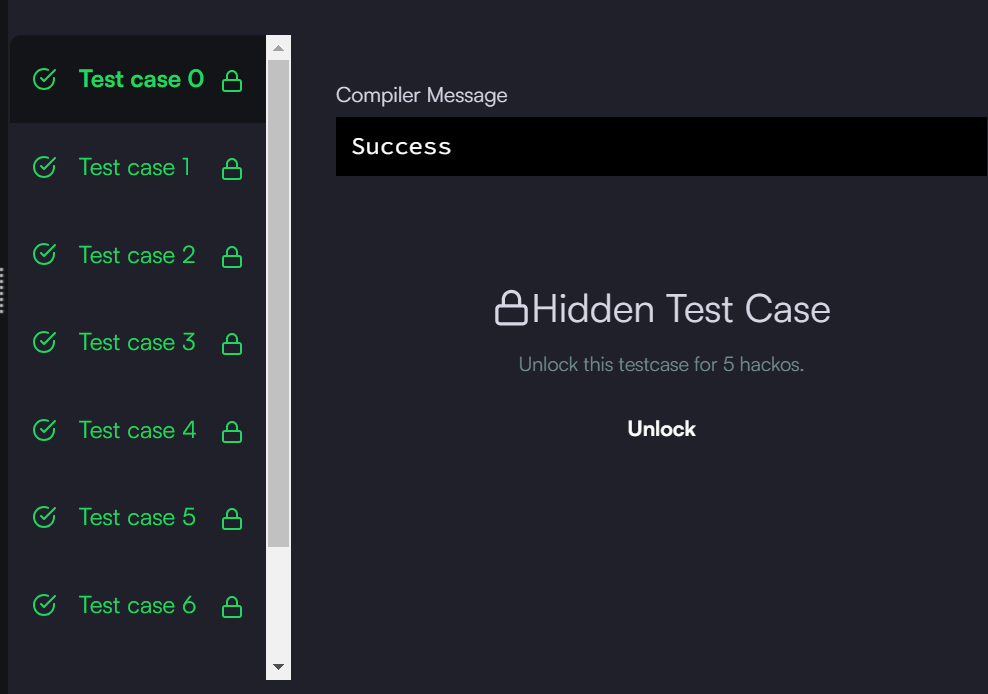
        break;

        }

    }

  return 0;

}



1. **Inheritance Introduction**

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

class Triangle{

    public:

        void triangle(){

            cout<<"I am a triangle\n";

        }

};

class Isosceles : public Triangle{

    public:

        void isosceles(){

            cout<<"I am an isosceles triangle\n";

        }

*//Write your code here.*

        void description() {

            cout << "In an isosceles triangle two sides are equal\n";

        }

};

int main(){

    Isosceles isc;

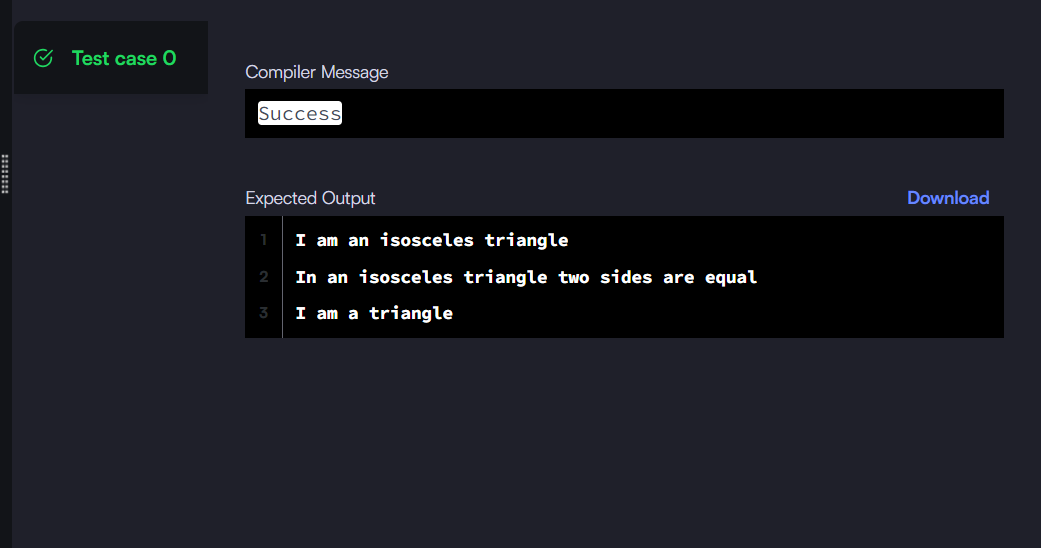
    isc.isosceles();

    isc.description();

    isc.triangle();

    return 0;

}

****

1. **Rectangle Area**

#include <iostream>

using namespace std;

*/\**

*\* Create classes Rectangle and RectangleArea*

*\*/*

class Rectangle{

    public:

    int height,width;

    void display(){

        cout<<width<<" "<<height<<endl;

    }

};

class RectangleArea: public Rectangle{

    public:

    void read\_input(){

        cin>>width>>height;

    }

    void display(){

        cout<<(width\*height);

    }

};

int main()

{

*/\**

*\* Declare a RectangleArea object*

*\*/*

    RectangleArea r\_area;

*/\**

*\* Read the width and height*

*\*/*

    r\_area.read\_input();

*/\**

*\* Print the width and height*

*\*/*

    r\_area.Rectangle::display();

*/\**

*\* Print the area*

*\*/*

    r\_area.display();

    return 0;

}

