You can view this report online at : https://www.hackerrank.com/x/tests/1665316/candidates/56942980/report

Full Name:	Breeha Qasim
Email:	bq08283@st.habib.edu.pk
Test Name:	**CS224 Lab# 07 - Fall 2023**
Taken On:	11 Oct 2023 12:39:25 PKT
Time Taken:	4299 min 45 sec/ 8640 min
Work Experience:	< 1 years
Invited by:	Shayan
Skills Score:	
Tags Score:	

100%

190/190

scored in **CS224 Lab# 07 - Fall 2023** in 4299 min 45 sec on 11 Oct 2023 12:39:25 PKT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here - https://www.hackerrank.com/x/tests/1665316/candidates/56942980/report

	Question Description	Time Taken	Score	Status
Q1	Complex > Coding	1 hour 15 min 26 sec	80/ 80	✓
Q2	Time > Coding	25 min 47 sec	60/ 60	!
Q3	Area Calculator > Coding	8 hour 25 min 13 sec	50/ 50	✓

QUESTION 1

Correct Answer

Score 80

Complex > Coding

QUESTION DESCRIPTION

Write a class called *Complex* to model a complex number, i.e. to store the real and imaginary part of the complex number (as a private attributes).

Write methods to overload the addition, subtraction, multiplication, cin>> and cout<< operators so that the code in main() works.

Example Test case:
Input

```
2 4
5 9
```

`Output`

```
3 + 5i
7 + 13i
-26 + 38i
```

First complex number is 2+4i, Second number is 5+9i
output is calculated as:
c2 - c1 ==> 3 + 5i
c1+c2 ==> 7 + 13i
c1*c2 ==> -26 + 38i

INTERVIEWER GUIDELINES

Solution

```
class Complex {
private:
  int m_real, m_imag;

public:
  Complex(int x, int y) :m_real(x), m_imag(y)
  {}

  Complex(const Complex& other) : m_real(other.m_real),
m_imag(other.m_imag)
  {}

  Complex& operator=(const Complex& other) {
    if (this != &other) {
      m_real = other.m_real;
      m_imag = other.m_imag;
    }
    return *this;
  }

  Complex operator+(const Complex& other) const {
    return Complex(m_real + other.m_real, m_imag + other.m_imag);
  }

  Complex operator-(const Complex& other) const {
    return Complex(m_real - other.m_real, m_imag - other.m_imag);
  }

  Complex operator*(const Complex& other) const {
    int real = m_real*other.m_real - m_imag*other.m_imag;
    int imag = m_real*other.m_imag + m_imag*other.m_real;
    return Complex(real, imag);
  }

  int get_real() const {
    return m_real;
  }

  int get_imaginary() const {
    return m_imag;
  }

};

std::ostream& operator<<(std::ostream& out, const Complex& c)
{
  out << c.get_real();
  if (c.get_imaginary() >= 0) {
    out << " + " << c.get_imaginary() << "i";
```

```
    }
    else {
      out << " - " << -c.get_imaginary() << "i";
    }
    return out;
}
```

CANDIDATE ANSWER

Language used: **C++**

```
using namespace std;
class Complex{
    private:
        int m_real;
        int m_imaginary;
    public:
        Complex()
            : m_real{0}, m_imaginary{0}
        {
        }
        Complex(int real, int imag)
            : m_real{real}, m_imaginary{imag}
        {
        }
        friend std::istream& operator>> (std::istream& in, Complex& complex);
        friend std::ostream& operator<< (std::ostream& out, const Complex&
complex);
        Complex operator+(Complex c1)
        {
            Complex temp;
            temp.m_real=m_real+c1.m_real;
            temp.m_imaginary=m_imaginary+c1.m_imaginary;
            return temp;
        }

        Complex operator-(Complex c2)
        {
            Complex temp2;
            temp2.m_real=m_real-c2.m_real;
            temp2.m_imaginary=m_imaginary-c2.m_imaginary;
            return temp2;
        }
        Complex operator*(Complex c3)
        {
            Complex temp3;
            temp3.m_real=(m_real*c3.m_real)-(m_imaginary*c3.m_imaginary);
            temp3.m_imaginary=(m_real*c3.m_imaginary)+
(c3.m_real*m_imaginary);
            return temp3;
        }
};
std::istream& operator>> (std::istream& in, Complex& complex)
{
    in >> complex.m_real;
    in >> complex.m_imaginary;
    return in;
}
std::ostream& operator<< (std::ostream& out, const Complex& complex)
{
```

```
    out<<complex.m_real;
    if (complex.m_imaginary<0)
    {
        out<<" - "<<-complex.m_imaginary<<"i"<<endl;

    }
    else
    {
      out<<" + "<<complex.m_imaginary<<"i"<<endl;
    }
    return out;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	60	0.0583 sec	8.75 KB
Testcase 1	Easy	Sample case	Success	10	0.0372 sec	8.93 KB
Testcase 2	Easy	Hidden case	Success	10	0.0322 sec	8.8 KB

No Comments

QUESTION 2

Needs Review

Score 60

Time › Coding

QUESTION DESCRIPTION

Create a class called **Time** that has separate int member data for **hours**, **minutes**, and **seconds**. One constructor should initialize this data to 0, and another should initialize it to passed values. Another member function (**show**) should display it, in 23:59:59 format. There should be overloaded + operator that adds the time of the object passed, and returns a Time object. A main() program is given, where two initialized time objects are created. Then it adds the two initialized objects together, leaving the result in the third time object. Finally it displays the value of all the time objects.

If any number exceeds its limit, it should be carried forward. For example, if the seconds are more than 59, they should carry forward to minutes. If the hours are more than 23, they can be truncated, as it will carry forward to next day.

INTERVIEWER GUIDELINES

```
class Time {
    int hours, minutes, seconds;

    public:
        Time() {
            hours = 0; minutes = 0; seconds = 0;
        }
        Time(int h, int m, int s){
            hours = h; minutes = m; seconds = s;
            if (s > 59) {
                m += (s / 60);
                s = s % 60;
            }
            if (m > 59) {
                h += (m / 60);
                m = m % 60;
            }
            if (h > 23) {
                h = h % 24;
            }
```

```
            if (h <= 23 && m <= 59 && s <= 59){
                hours = h; minutes = m; seconds = s;
            }
        }

        void show() {
            if (hours < 10){
                cout << 0;
            }
            cout << hours << ":";
            if (minutes < 10){
                cout << 0;
            }
            cout << minutes << ":";
            if (seconds < 10) {
                cout << 0;
            }
            cout << seconds << endl;
        }

        Time operator + (const Time& t1) {
            Time t3(hours + t1.hours, minutes + t1.minutes, seconds +
t1.seconds);
            return t3;
        }
};
```

CANDIDATE ANSWER

Language used: **C++**

```
#include <iomanip>
class Time
{
    private:
        int hours;
        int minutes;
        int seconds;
    public:
        Time()
            : hours{0}, minutes{0}, seconds{0}
        {
        }
        Time(int hr, int min, int sec)
            :hours{hr}, minutes{min}, seconds{sec}
        {
            if (seconds>=60)
            {
                minutes+=seconds/60;
                seconds=seconds%60;
            }
            if (minutes>=60)
            {
                hours+=minutes/60;
                minutes=minutes%60;
            }
            if (hours>24)
            {
                hours=hours%24;
            }
        }
        Time operator+(Time time)
```

```
        {
            Time temp;
            temp.hours=hours+time.hours;
            temp.minutes=minutes+time.minutes;
            temp.seconds=seconds+time.seconds;
            if (temp.seconds>=60)
            {
                temp.minutes+=temp.seconds/60;
                temp.seconds=temp.seconds%60;
            }
            if (temp.minutes>=60)
            {
                temp.hours+=temp.minutes/60;
                temp.minutes=temp.minutes%60;
            }
            if (temp.hours>24)
            {
                temp.hours=temp.hours%24;
            }
            return temp;
        }
        void show(){
            cout<<setw(2)<<setfill('0')<<hours<<":"<<setw(2)<<setfill('0')
<<minutes<<":"<<setw(2)<<setfill('0')<<seconds<<endl;
        }
};
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.022 sec	8.91 KB
Testcase 1	Easy	Sample case	Success	10	0.0235 sec	8.84 KB
Testcase 2	Easy	Sample case	Success	10	0.0519 sec	8.77 KB
Testcase 3	Easy	Sample case	Success	10	0.0221 sec	8.6 KB
Testcase 4	Easy	Sample case	Success	10	0.0227 sec	8.98 KB
Testcase 5	Easy	Sample case	Success	10	0.034 sec	8.87 KB

No Comments

QUESTION 3

Correct Answer

Score 50

Area Calculator › Coding

QUESTION DESCRIPTION

Create a class called **Area** that uses two variables (length, and width) of type **Distance** to model the area of a rectangle. Initialize a variable of type **Area** to specific dimensions (taken as input). Overload the double operator, so that Area object can be converted to double value, returning area in square feet.

To calculate the area, convert each dimension from a Distance variable to a variable of type double representing feet and fractions of a foot, and then multiply the resulting two numbers.

▼ Details

▼ Sample Case 0

Sample Input For Custom Testing

```
5 3
2 6
```

Sample Output

```
Area is: 13.125sq feet
```

Explanation

First line of input is length: 5'3"

Second line of input is width: 2'6"

Area is calculated by converting length, and width to feet in decimal. So, length is converted to 5.25, width is converted to 2.5.

Finally 5.25*2.5 = 13.125.

▼ Details

INTERVIEWER GUIDELINES

```
#include<iostream>
using namespace std;

class Area
{
    Distance length, width;

    public:
        Area()
        {}

        Area(Distance l, Distance w)
        {
            length = l;
            width = w;
        }

        operator double()
        {
            double d1 = length.feet + length.inches/12.0;
            double d2 = width.feet + width.inches/12.0;
            double ar = d1 * d2;
            return ar;
        }
};
```

CANDIDATE ANSWER

Language used: **C++**

```
1
2  class Area
3  {
4      private:
5          Distance length;
6          Distance distance;
7      public:
8          Area(const Distance& len,const Distance& d)
9              :  length{len}, distance{d}
10         {
11         }
12         operator double() const
13         {
14             double length_to_ft=length.feet + length.inches/12.0;
15             double width_to_ft=distance.feet + distance.inches/12.0;
16             return length_to_ft*width_to_ft;
17         }
```

```
};

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0287 sec	8.92 KB
Testcase 1	Easy	Sample case	Success	10	0.0232 sec	8.88 KB
Testcase 2	Easy	Sample case	Success	10	0.0361 sec	8.92 KB
Testcase 3	Easy	Sample case	Success	10	0.0535 sec	8.73 KB
Testcase 4	Easy	Sample case	Success	10	0.0233 sec	8.89 KB

No Comments

PDF generated at: 19 Oct 2023 10:09:45 UTC