CSDC202 - Programming Assignment No 1

General Instructions

Solve all of the programming problems given. Write correct and properly documented C++ codes.

Grading

The maximum points for this assignment is 100. Each working and properly documented C++ program that displays correct output (using the teacher's test set) merits twenty-five (25) points.

Any instance of plagiarism found on any source code submitted merits a grade of zero (0) for this whole assignment. Late submissions will incur a deduction of one (1) point per five (5) minutes of lateness.

Documentation

Put comments in your source code explaining what each statement or group of statements do.

Also, complete and include the header info and honor code shown below.

```
#-------------------------------------------------------------------------#
# Filename        :                                                       #
# Description     :                                                       #
# Author          :                                                       #
# Course and Year :                                                       #
# Last Modified   :                                                       #
# Honor Code      : This is my own code. I have worked hard in completing #
#                   this work and I have not copied from any unauthorized #
#                   resource. I am also well aware of the policies        #
#                   stipulated in the college student handbook regarding  #
#                   academic dishonesty.                                  #
#-------------------------------------------------------------------------#
```

Should you consulted any reference (including but not limited to books, journal articles, blogs, etc), cite them properly in your source code. Explain up to what extent the help you derive from any particular reference.

Submission and Deadline

All source files must be submitted on or before July 29, 2019 at 8AM via DCS Moodle (https://courses.dcs.adnu.edu.ph). No need to zip your source codes.

Problem A: Triangles

Source File	*yoursurname*_triangles.cpp

Problem Description

Write a program that prints a triangle shape composed of specific ASCII character.

Input Format

Input is read from STDIN and consists of multiple lines. The first line contains the number of test cases, T. This is then followed by T lines containing an integer k, which denotes the height of the triangle that you need to build, and a character c, which will be used in building the shape. Assume that c is not a whitespace character.

Output Format

Output must be displayed in STDOUT. The output for each case begins with a line containing the word CASE followed by a single space, the case number starting from 1, and a colon (:). This is followed by k lines corresponding to each level of the triangle shape with height k.

Sample Input

```
3
1 @
2 #
5 $
```

Sample Output

```
CASE 1:
@
CASE 2:
 #
###
CASE 3:
    $
   $$$
  $$$$$
 $$$$$$$
$$$$$$$$$
```

Problem B: Fibonacci

Source File	*yoursurname*_fibonacci.cpp

Problem Description

The Fibonacci sequence is the series 0,1,1,2,3,5,8,13,... where the next number is found by adding the numbers before it. More formally, the sequence F_n of Fibonacci numbers is defined by the recurrence relation:

$F_n = F_{n-1} + F_{n-2}$ with initial values of $F_0 = 0$ and $F_1 = 1$.

Write a program that reads an integer n and prints the n^{th} Fibonacci number, F_n. Your program must define the function **fib()** with an integer parameter n and computes and returns F_n.

Input Format

Input is read from STDIN and consists of multiple lines. The first line contains the number of test cases, T. This is then followed by T lines consisting a single integer $n \geq 0$.

Output Format

Output must be displayed in STDOUT. The output for each case begin with the word CASE followed by a single space, the case number starting from 1, a colon (:), and a single space and the n^{th} Fibonacci number F_n.

Sample Input

```
2
5
10
```

Sample Output

```
CASE 1: 5
CASE 2: 55
```

Problem C: Digits

Source File	*yoursurname*_digits.cpp

Problem Description

Write a program that reads an integer and “separates” this integer into its individual digits then computes and displays the sum of these digits.

Your program must define the function

int split_and_sum (int);

which takes in an integer as an argument and computes and returns the sum of the individual digits comprising the argument.

Input Format

Input is read from STDIN and consists of multiple lines. The first line contains the number of test cases, T. This is then followed by T lines consisting a single integer between 1 and 32767.

Output Format

Output must be displayed in STDOUT. The output for each case begin with the word CASE followed by a single space, the case number starting from 1, a colon (:), and a single space. Then, the sum of the digits of the corresponding integer input.

Sample Input

```
2
56
3456
```

Sample Output

```
CASE 1: 11
CASE 2: 18
```

Problem D: Complex Numbers

Source Files	*yoursurname*_complex.h *yoursurname*_complex.cpp *yoursurname*_testcomplex.cpp

Problem Description

Extend the functionality of the Complex class to be able to support addition, subtraction and multiplication of Complex numbers. The source code for the Complex class is uploaded in our course site (See Code Examples, C++ Review Pt 1). Note that to complete this problem successfully, you must now how to overload the C++ +, -, and * operators.

Refer to the table below for the rules in computing the sum, difference and product of complex numbers:

Rule	**Example**	**Remarks**
(a + b*i*) + (c + d*i*) = (a + c) + (b + d)*i*	(1 + *i*) + (6 - 5*i*) = 7 - 4*i*	To add complex numbers, add up the real parts (without *i*) and add up the imaginary parts (with *i*)
(a + b*i*) - (c + di) = (a - c) + (b - d)*i*	(1 + *i*) - (3 - 5*i*) = -2 + 6*i*	To subtract complex numbers, subtract the real parts and subtract the imaginary parts (with *i*)
(a + b*i*) * (c + d*i*) = (ac - bd) + (ad + bc)*i*	(1 + *i*) * (3 + 5*i*) = 1*3 + 1*5*i* + *i**3 + *i**5*i* = 3 + 5*i* + 3*i* - 5 = -2 + 8*i*	To multiply complex numbers, use the distributive law, simplify the resulting polynomial and apply $i^2 = -1$.

After completely implementing the additional methods for the Complex class, write a program (*yoursurname*_testcomplex.cpp) to test your implementation. Your test program must contain the following declarations:

```
Complex *v = new Complex (1, -2);
Complex *w = new Complex (3, 4);
Complex *x = new Complex (0, -5);
Complex *y = new Complex (0, 5);
```

Finally, using the above declarations, your test program must display the sum, difference and product of the complex numbers pointed to by the pair of pointers v and w, w and x, and x and y.

Input Format

This problem does not require any input.

Output Format

Output must be displayed in STDOUT. The first three (3) lines represent the sum, difference and product of the complex numbers pointed to by v and w. The next three (3) lines represent the sum, difference and product of the complex numbers pointed to by w and x. Finally, the last three (3) lines represent the sum, difference and product of the complex numbers pointed to by x and y.

Expected Output

```
4 + 2i
-2 - 6i
11 - 2i
3 - i
3 + 9i
20 - 15i
0
10i
25
```