American Computer Science League

2021-2022 ● Contest 4: Fibonacci & Mandelbrot ● Senior Division

PROBLEM: Given the function $f(z) = z^2 + C$, "iterate" the function by evaluating $f(f(f(...f(f(0)))))$, stopping as soon as the function *escapes* or *cycles*. The function *escapes* when its absolute value is greater than a specified value; in this problem it is 4. The function *cycles* when it results in a value that it has already produced after rounding each result to 3 decimal places. The variable z and the constant C are *complex* numbers. You'll be evaluating a set of values of C and reporting information about the evaluation of f.

In order to use *complex* numbers, we need to define an *imaginary* number, i, such that $i = \sqrt{-1}$, and therefore, the value of $i^2 = \sqrt{-1} * \sqrt{-1} = -1$. A *complex* number has two parts: a *real* part and an *imaginary* part. They are represented in the form $a + bi$, where a is referred to as the *real* number part and bi as the *imaginary* number part. Both *a* and *b* are *real* numbers.

There are three operations you will need to perform on *complex* numbers: addition, multiplication, and absolute value. Here is how each operation is performed:

- **Addition.** Combine the *real* number parts and combine the *imaginary* number parts:

  $$(a + bi) + (c + di) = (a + c) + (b + d)i$$

  Example: $(5 + 2i) + (3 - 6i) = 8 - 4i$

- **Multiplication.** Use the distributive property on binomial terms:

  $$\begin{aligned}(a + bi)(c + di) &= ac + adi + bci + bdi^2 \\ &= (ac - bd) + (ad + bc)i\end{aligned}$$

  Example:
  $$\begin{aligned}(5 + i)(3 + 2i) &= 5 \cdot 3 + 5 \cdot 2i + 1 \cdot 3i + 1 \cdot 2i^2 \\ &= 15 + 10i + 3i - 2 \\ &= 13 + 13i\end{aligned}$$

  Example: $(5 + 2i)(3 - 6i) = 27 - 24i$

- **Absolute Value.** The absolute value of a *complex* number is given by the formula:

  $$|a + bi| = \sqrt{a^2 + b^2}$$

  Example: $|3 - 4i| = \sqrt{3^2 + (-4)^2} = \sqrt{9 + 16} = \sqrt{25} = 5.$

Here are two examples that show how to evaluate f. In the first example, the function *escapes* because the evaluation stops if the absolute value of the function is greater than 4. In the second example, the function *cycles* and evaluation stops because the value of the function, always rounded to three decimal places, has already appeared. When rounding, find the closest value to the calculated result with only 3 decimal places and if there's a 5 in the 4th decimal place, round away from 0. This is the default for the *round* function in Java, C++, and Python. We guarantee that there will be no more than 500 iterations before the function either *cycles* or *escapes*.

EXAMPLE #1: $C = 2 - 0.3i$

$f(0) = (0)^2 + (2 - 0.3i) = 2 - 0.3i$ (This absolute value < 4.)

$f(f(0)) = f(2 - 0.3i) = (2 - 0.3i)^2 + (2 - 0.3i)$

$= (4 - 0.6i - 0.6i + 0.09i^2) + (2 - 0.3i)$

$= (4 - 0.09 + 2) + (-0.6 - 0.6 - 0.3)i$

$= 5.91 - 1.5i$

The absolute value of $5.91 - 1.5i = \sqrt{5.91^2 + (-1.5)^2} = \sqrt{34.9281 + 2.25} = 6.09738$. This value is larger than 4, so the evaluation stops.

EXAMPLE #2: $C = -0.065 - 0.2i$

$f(0) = (0)^2 + (-0.065 - 0.2i) = -0.065 - 0.2i$

$f(f(0)) = f(-0.065 - 0.2i) = (-0.065 - 0.2i)^2 + (-0.065 - 0.2i) = -0.101 - 0.174i$

$f(f(f(0))) = f(-0.101 - 0.174i) = (-0.101 - 0.174i)^2 + (-0.065 - 0.2i) = -0.085 - 0.165i$

$f(f(f(f(0)))) = f(-0.085 - 0.165i) = (-0.085 - 0.165i)^2 + (-0.065 - 0.2i) = -0.085 - 0.172i$

$f(f(f(f(f(0))))) = f(-0.085 - 0.172i) = (-0.085 - 0.172i)^2 + (-0.065 - 0.2i) = -0.087 - 0.171i$

$f(f(f(f(f(f(0)))))) = f(-0.087 - 0.171i) = (-0.087 - 0.171i)^2 + (-0.065 - 0.2i) = -0.087 - 0.17i$

$f(f(f(f(f(f(0)))))) = f(-0.087 - 0.17i) = (-0.087 - 0.17i)^2 + (-0.065 - 0.2i) = -0.086 - 0.17i$

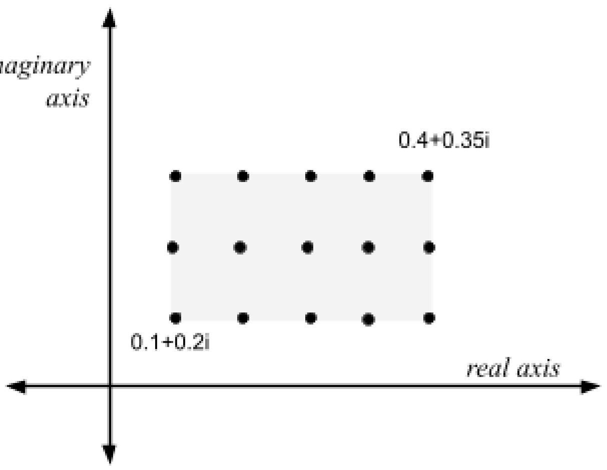
$f(f(f(f(f(f(0)))))) = f(-0.086 - 0.17i) = (-0.086 - 0.17i)^2 + (-0.065 - 0.2i) = -0.087 - 0.171i$

Since the absolute value never exceeds 4 and this last answer is the same value as the 5th result, the evaluation stops. The length of the resulting *cycle* is 3; that is, the function would *cycle* through 3 different values even though other values preceded them.

In this program, you will be given 2 complex numbers and an increment amount. These numbers define a rectangular area in the *complex* plane and points of interest in that area. The *complex* number $a + bi$ is plotted in the *complex* plane as *(a, b)* with *a* along the *real* axis and *b* along the *imaginary* axis. For example, "0.1 0.2 0.4 0.35 0.075" indicates that the points (0.1, 0.2) and (0.4, 0.35) are the lower-left and the upper-right corners of the rectangular area respectively and the increment amount is 0.075. Every value being checked should initially be rounded to 3 decimal places (both the *real* part and the *imaginary* part) to avoid roundoff error with floating point numbers. We guarantee that the increment will be smaller than either range of values and will include the points on the perimeter in every case. In this example, the increment amount added to both *a* and *b* from the lower-left corner to upper-right corner of the rectangular area will generate the following 15 *complex* numbers:

(0.1+0.35*i*)	(0.175+0.35*i*)	(0.25+0.35*i*)	(0.325+0.35*i*)	(0.4+0.35*i*)
(0.1+0.275*i*)	(0.175+0.275*i*)	(0.25+0.275*i*)	(0.325+0.275*i*)	(0.4+0.275*i*)
(0.1+0.2*i*)	(0.175+0.2*i*)	(0.25+0.2*i*)	(0.325+0.2*i*)	(0.4+0.2*i*)

Graphically, the 15 *complex* numbers are plotted as follows:



For each *complex* number that is generated, iterate the function until a *cycle* is found or the function *escapes*. At each step, round the result to the nearest one-thousandth (both the *real* part and the *imaginary* part). For functions that do not result in an *escape*, a *cycle* will eventually emerge. Record the length of the *cycle*. Output the number of generated *complex* numbers having a *cycle* length that is a Fibonacci number (1, 2, 3, 5, 8, 13, 21, 34, ….) less than 500.

EXAMPLE #3: Input 0.1 0.2 0.4 0.35 0.075

The following values of *C* are evaluated. The bold number to the right is 0 if the function *escapes*, or a positive number indicating the length of the *cycle*.

(0.1+0.2*i*) **1**	(0.175+0.2*i*) **1**	(0.25+0.2*i*) **1**	(0.325+0.2*i*) **6**	(0.4+0.2*i*) **0**
(0.1+0.275*i*) **1**	(0.175+0.275*i*) **1**	(0.25+0.275*i*) **1**	(0.325+0.275*i*) **6**	(0.4+0.275*i*) **0**
(0.1+0.35*i*) **1**	(0.175+0.35*i*) **1**	(0.25+0.35*i*) **4**	(0.325+.035*i*) **5**	(0.4+0.35*i*) **105**

There are 2 values of *C* that *escape*. Of the 13 other *C* values, 9 of them have *cycles* whose lengths are Fibonacci numbers.

INPUT: Each data set has 5 decimal values representing the lower-left corner and the upper-right corner respectively of a rectangular region on the *complex* plane. Both *complex* numbers represent a point *(a,b)* for a *complex* number a+b*i*. The 5th number is the increment between the points on both the *real* axis and the *imaginary* axis.

OUTPUT: For each line of data, find the total number of *C* values that have a *cycle* length that is equal to one of the numbers in the Fibonacci sequence.

SAMPLE INPUT:

```
0.1 0.2 0.4 0.35 0.075
-0.1 -0.1 0.0 0.1 0.05
-2.5 -1.0 -2.0 0.0 0.05
-0.1 -0.3 -0.05 -0.2 0.005
0.13 -0.3 0.2 -0.2 0.01
```

SAMPLE OUTPUT:

```
  1. 9
  2. 15
  3. 1
  4. 222
  5. 72
```

TEST DATA

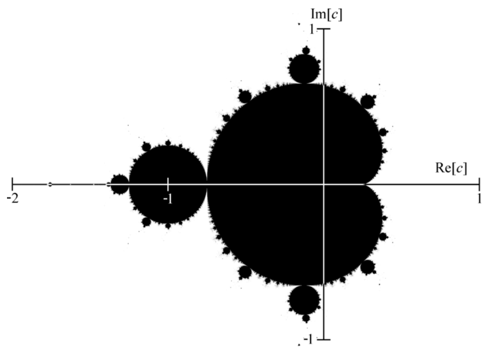
TEST INPUT:

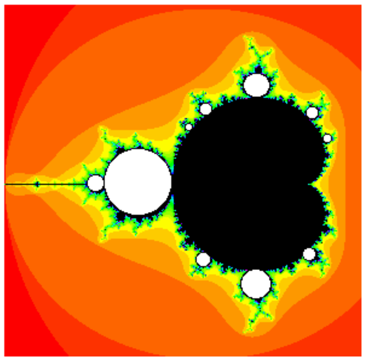
```
-1.5 -1.5 1.5 1.5 0.3
-0.4 -0.3 0.4 0.3 0.06
0.175 -0.1 0.235 0.4 0.006
-0.2 -0.2 0.0 0.5 0.02
-0.375 -0.1 0.075 0.5 0.025
```

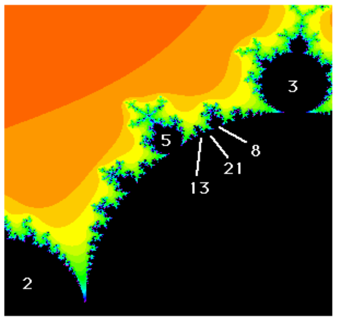
TEST OUTPUT:

```
  1. 15
  2. 127
  3. 764
  4. 388
  5. 456
```

BACKGROUND INFORMATION FOR THE ADVISOR:







The Mandelbrot Set is generated by iterating the function $f(z) = z^2 + C$ for points within a small range on the complex plane where the point (a,b) represents the complex number a + bi as shown in the 1st illustration.

When iterating a function over itself, the function either converges to a single value, cycles between multiple values, or “escapes” to infinity, meaning that its value gets larger each time. If you are given an initial value for C, a complex number, starting with a value of z = 0, the function generates a sequence of numbers until one of these occurs.

If the iteration results in converging or cycling among 2 or more values, the point is colored back because it is in the Mandelbrot Set. The various colors outside the Mandelbrot Set are used to illustrate how many iterations it takes to exceed some value, thus how quickly the values “escape” to infinity.

The main “bulb” (black area in the 2nd illustration) in the Mandelbrot Set represents all points that converge or have a cycle of length 1. The smaller bulbs (white areas in the 2nd illustration) represent points with different cycle lengths.

The Fibonacci sequence can be found in the Mandelbrot Set as the length of the cycle for all of the points in each of the bulbs in the upper left and lower left sections of the main bulb as shown in the 3rd illustration given.