American Computer Science League

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

PROBLEM: Given the function $f(z) = z^2 + C$, "iterate" the function by evaluating $f(f(f(\ldots f(f(0)))))$, stopping as soon as the function *escapes* or *cycles*. The function *escapes* when the absolute value of its result 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 2 decimal places. The variable z and the constant C are *complex* numbers.

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* part and bi as the *imaginary* 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:

  $$(a + bi)(c + di) = ac + adi + bci + bdi^2$$
  $$= (ac - bd) + (ad + bc)i$$

  Example: $(5 + i)(3 + 2i) = 5 \cdot 3 + 5 \cdot 2i + 1 \cdot 3i + 1 \cdot 2i^2$
  $$= 15 + 10i + 3i - 2$$
  $$= 13 + 13i$$

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

In this program, round the result of each iteration to the nearest hundredth (both the *real* part and the *imaginary* part). When the function result matches a previous value, output the length of the cycle, (i.e., the number of unique values from the first of the matching values to the second one). If the absolute value is greater than 4, output "ESCAPES *n*", where *n* is the number of the last iteration that was completed. There is a single space between the word ESCAPES and the number *n*. When rounding, find the closest value to the calculated result with only 2 decimal

places and if there's a 5 in the third 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 = -0.1 + 0.75*i*

$f(0) = (0 + 0i)^2 + (-0.1 + 0.75i) = -0.1 + 0.75i$

$f(-0.1 + 0.75i) = (-0.1 + 0.75i)^2 + (-0.1 + 0.75i) = -0.65 + 0.6i$

$f(-0.65 + 0.6i) = (-0.65 + 0.6i)^2 + (-0.1 + 0.75i) = -0.04 - 0.03i$

$f(-0.04 - 0.03i) = (-0.04 - 0.03i)^2 + (-0.1 + 0.75i) = -0.1 + 0.75i$

Since none of these absolute values is greater than 4, and this last result is the same as the first result, this is a cycle of 3 values. The output is "3".

EXAMPLE #2: C = 2 - 0.3*i*

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

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

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

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

$= 5.91 - 1.5i$

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 an input value of 2 - 0.3*i* would output "ESCAPES 2".

INPUT: Two real numbers representing the *real* part and the *imaginary* part of the *complex* number *C*, in the function $f(z) = z^2 + C$**.**

OUTPUT: For each input, output a string that is either the length of the cycle if the function *cycles* or the string "ESCAPES *n*" if the function *escapes*.

SAMPLE INPUT:

```
-0.1 0.75
2.0 -0.3
-0.5 0.56
-1.21 -0.32
0.01 -0.64
```

SAMPLE OUTPUT:

```
1. 3
2. ESCAPES 2
3. 5
4. 8
5. 13
```

American Computer Science League

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

TEST DATA

TEST INPUT:

```
-0.52 0.57
-1.07 -0.2
-1.04 -0.28
0.33 0.77
0.26 -0.02
```

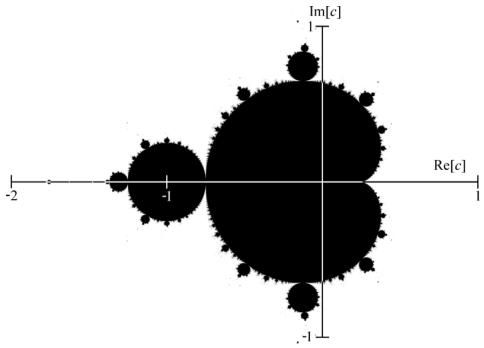
TEST OUTPUT:

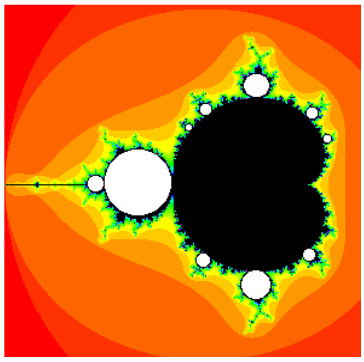
1. 5
2. 8
3. 21
4. ESCAPES 6
5. 1

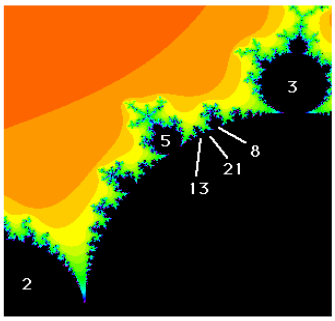
American Computer Science League

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

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.

American Computer Science League

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

问题: 给定一个函数 $f(z) = z^2 + C$, 通过反复“迭代”函数 $f(z)$，计算 $f(f(f(\ldots f(f(0)))))$ 的值。一旦函数 $f(z)$ 遇到“escapes”或者“cycles”现象，便停止计算。当函数结果的绝对值大于指定的值，表明函数 $f(z)$ 发生了“escapes”现象（这个题目中，指定值是 4）。当函数 $f(z)$ 产生一个值，且生成的每个结果都四舍五入到小数点后 2 位，表明函数 $f(z)$ 发生了“cycles”现象。变量 z 和常数 C 是复数。

为了使用复数，我们需要定义一个虚数, i, 为此，我们定义 $i = \sqrt{-1}$，于是可以得到 $i^2 = \sqrt{-1} * \sqrt{-1} = -1$。一个复数有两个部分：实部和虚部，以 $a + bi$ 的形式表示。其中 a 表示实部，bi 表示虚部。a 和 b 都为实数。

你需要对复数执行三种运算：加法、乘法和取绝对值。以下是每种运算的执行方式：

- **加法.** 实部和实部相加，虚部和虚部相加：

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

  例如: $(5 + 2i) + (3 - 6i) = 8 - 4i$

- **乘法.** 使用二项式乘法分配律：

  $$(a + bi)(c + di) = ac + adi + bci + bdi^2$$
  $$= (ac - bd) + (ad + bc)i$$

  例如: $(5 + i)(3 + 2i) = 5 \cdot 3 + 5 \cdot 2i + 1 \cdot 3i + 1 \cdot 2i^2$
  $$= 15 + 10i + 3i - 2$$
  $$= 13 + 13i$$

  例如: $(5 + 2i)(3 - 6i) = 27 - 24i$

- **绝对值.** 根据以下公式得出一个复数的绝对值：

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

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

在这个程序中，将每次迭代的结果四舍五入到其最接近的百分位（实部和虚部）。当函数结果与先前输出的值匹配时，输出循环的长度（即，从第一个匹配值到第二个匹配值的唯一值的数量）。如果绝对值大于 4，则输出 “ESCAPES *n*”，其中 *n* 是已完成的最后一次迭代的次数。单词 ESCAPES 和数字 *n* 之间只有一个空格。四舍五入时，找到与小数点后两位的计算结果最接近的值。

如果小数点后第三位是 5，则从 0 开始四舍五入。这是 Java, C++, 和 Python 中四舍五入函数的默认方式。我们保证在函数发生 “ cycles ” 现象或者 “ escapes ” 现象前，函数的迭代次数不会超过 500 次。

例如 #1: C = -0.1 + 0.75*i*

$f(0) = (0 + 0i)^2 + (-0.1 + 0.75i) = -0.1 + 0.75i$

$f(-0.1 + 0.75i) = (-0.1 + 0.75i)^2 + (-0.1 + 0.75i) = -0.65 + 0.6i$

$f(-0.65 + 0.6i) = (-0.65 + 0.6i)^2 + (-0.1 + 0.75i) = -0.04 - 0.03i$

$f(-0.04 - 0.03i) = (-0.04 - 0.03i)^2 + (-0.1 + 0.75i) = -0.1 + 0.75i$

因为这些绝对值都不大于 4，并且最后一个结果和第一个结果相同，这是一个由 3 个值组成的循环。输出为 “3”。

例如 #2: C = 2 - 0.3*i*

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

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

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

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

$= 5.91 - 1.5i$

绝对值 $5.91 - 1.5i = \sqrt{5.91^2 + (-1.5)^2} = \sqrt{34.9281 + 2.25} = 6.09738.$

该值大于 4，所以输入值 2 - 0.3*i* 会输出 “ ESCAPES 2” 。

输入：两个实数，分别代表函数 $f(z) = z^2 + C$ 中复数 C 的实部和虚部。

输出：对于**每个输入，会输出一个字符串：如果函数**发生 “ cycles ” 现象，**则**输出**循环长度；如果函数**发生 “ escapes ” 现象，**则**输出**字符串 “ ESCAPES *n* ”。**

样本输入：

```
-0.1 0.75
2.0 -0.3
-0.5 0.56
-1.21 -0.32
0.01 -0.64
```

样本输出：

```
1. 3
2. ESCAPES 2
3. 5
4. 8
5. 13
```

American Computer Science League

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

测试数据

测试输入:

```
-0.52 0.57
-1.07 -0.2
-1.04 -0.28
0.33 0.77
0.26 -0.02
```

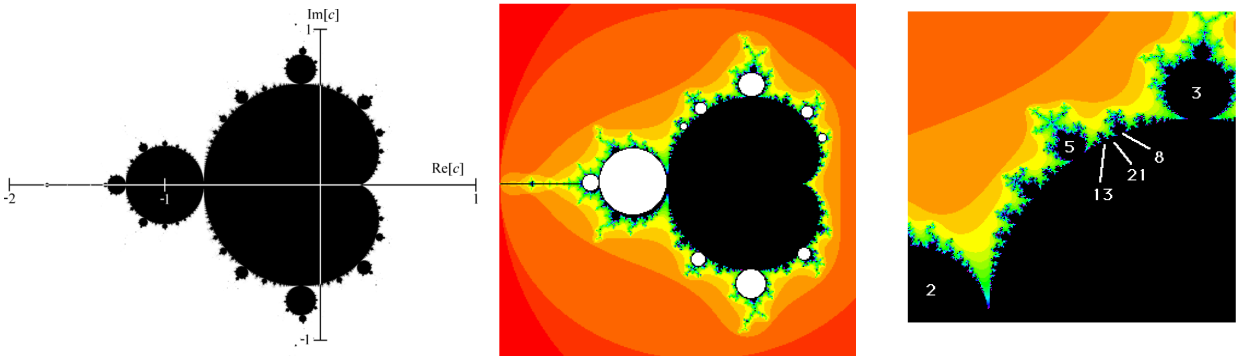
测试输出:

1. 5
2. 8
3. 21
4. ESCAPES 6
5. 1

American Computer Science League

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

背景信息:



曼德布洛特**集合是通过迭代函数** $f(z) = z^2 + C$**，**在复**平面上**输出**小**范围内**的点**而生成的，**其中点（**a , b**）表示复数** $a + bi$**，如图 1 所示。**

当函数进行**自**我**迭代时，函数要么收敛**至**一个值，在多个值之间**“ 反复**循环 ”，要么**“ escapes ”到无穷大，这意味着它的值每次都会变大。如果给你一个复数** C **的初始值，**且**从** $z = 0$ **开始，**该**函数会生成一**系列**数字，直到**出现上述现象中的**一个。**

如果迭代的结果是在 2 个或多个值之间收敛或循环，则该点的背景会被着色，因为它位于曼德布洛特**集合中。曼德**布洛特**集合外的各种颜色是用**来**说明需要多少次迭代才能超过某个值，从而说明这个值“ escapes ”到无穷大的速度**有多快。

曼德布洛特**集合中的“ 主**灯泡区域 ”**（第二幅图中的黑色区域）表示所有收敛或循环长度为 1 的点。较小的“** 灯泡区域 ”**（第二幅图中的白色区域）代表具有不同循环长度的点。**

如给出的第 3 幅图所示，在“ 主灯泡区域 ”的左上部分和左下部分中，每个“ 灯泡区域 ”中表示循环长度的所有点都能够在曼德布洛特集合中找到斐波那契数列。