Problem 106: Mandelbrot Set

Difficulty: Medium

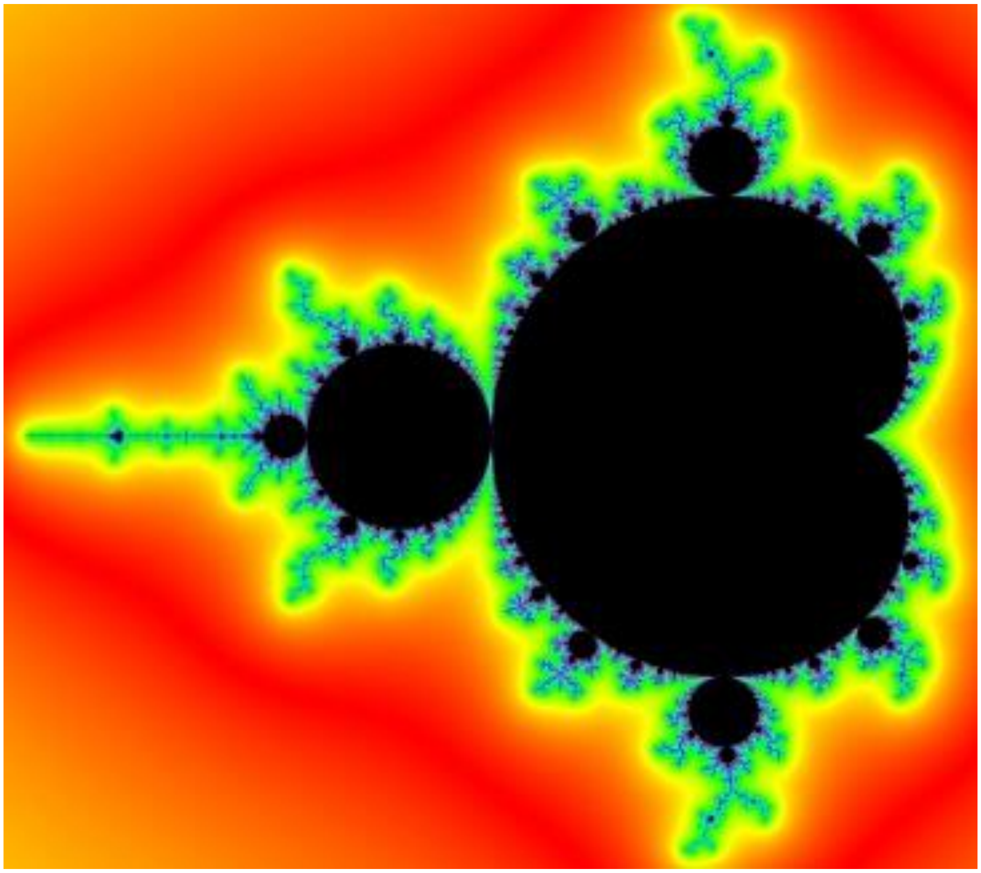
Author: Louis Ronat, Denver, Colorado, United States

Originally Published: Code Quest 2019

Problem Background

The Mandelbrot set is drawn by considering the recursive function $Z_{n+1} = {Z_n}^2 + c$, where c is a complex number of the form $a + bi$ (in mathematics, i is an imaginary number with the value of $\sqrt{-1}$; thus, $i^2 = -1$). By iterating repeatedly, using each value of Z to calculate the next value, we find that for some input values of c, Z grows without bound. For others, Z remains bound.

To draw the Mandelbrot set, we use the "complex plane", where the horizontal x-axis represents the value of a, and the vertical y-axis represents the value of b. Each point is colored based on the number of iterations (n) we can perform before the absolute value of Z ($|Z_n|$) becomes greater than a specified value. When this happens, it is said that the function "diverges". In the image below, black indicates that $|Z_n|$ remained below a prescribed value for all values of n. Blue pixels represent points at which it took many iterations to get $|Z_n|$ above that value; red pixels required fewer iterations.



Let's consider the function using a value of $c = 1.1 + 2i$.

Regardless of the value of c, the value of Z_0 always equals 0. We can use this to determine the value of Z_1:

$$Z_1 = {Z_0}^2 + c$$
$$Z_1 = 0^2 + 1.1 + 2i$$
$$Z_1 = 1.1 + 2i$$

From this, we can see that for any value of c, $Z_1 = c$. Now we need to determine if the function has diverged. For the purposes of this problem, we'll consider the function to have diverged if $|Z_n| \geq 100$. Since i is an imaginary number, we use this formula to determine the absolute value of numbers of the form $a + bi$:

$$|Z_1| = \sqrt{{a_1}^2 + {b_1}^2}$$
$$|Z_1| = \sqrt{1.1^2 + 2^2}$$
$$|Z_1| = \sqrt{1.21 + 4}$$
$$|Z_1| \approx 2.2825$$

2.2825 is less than 100, so the function hasn't diverged yet. We need to do more iterations to determine when it diverges, if ever:

$$Z_2 = {Z_1}^2 + c$$
$$Z_2 = (a_1 + b_1 i)^2 + a_0 + b_0 i$$
$$Z_2 = (1.1 + 2i)^2 + 1.1 + 2i$$
$$Z_2 = 1.1^2 + 1.1(2i) + 1.1(2i) + (2i)^2 + 1.1 + 2i$$
$$Z_2 = 1.21 + 4.4i - 4 + 1.1 + 2i$$
$$Z_2 = -1.69 + 6.4i$$
$$a_2 = -1.69$$
$$b_2 = 6.4$$
$$|Z_2| = \sqrt{-1.69^2 + 6.4^2}$$
$$|Z_2| \approx \sqrt{2.8561 + 40.96}$$
$$|Z_2| \approx 6.6194$$

(Remember that $i^2 = -1$, so above, $(2i)^2 = 2^2 * i^2 = 4 * -1 = -4$.)

$|Z_2|$ is still less than 100, so it hasn't diverged yet. How many iterations do we need to do to reach that point?

n	*Z*	*a*	*b*	|*Z*|
1	1.1 + 2*i*	1.1	2	2.2825
2	-1.69 + 6.4*i*	-1.69	6.4	6.6194
3	-37.0039 - 19.632*i*	-37.0039	-19.632	41.8892
4	984.9732 + 1454.9211*i*	984.9732	1454.9211	1756.9769

So at $n = 4$, we see that the value of $|Z| > 100$. This means that for this value of c, the function has diverged at 4. We color the point at x = 1.1, y = 2 an appropriate color for that value, and move on to the next value of c to be checked.

Problem Description

Your program must identify the color to use in a rendering of the Mandelbrot set for a given value of c. Use the following table and the explanation above to determine what colors should be used:

Value of n when function diverges	Color
⩽ 10	RED
11-20	ORANGE
21-30	YELLOW
31-40	GREEN
41-50	BLUE
⩾ 51	BLACK

For the example calculation above, the function diverged at $n = 4$, so the color for that value of c should be red.

Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include a single line of input with two decimal numbers separated by spaces. These numbers represent the values for a and b, respectively. Remember that $c = a + bi$.

```
4
1.1 2.0
-0.7 0.2
-0.5 0.65
-0.5 0.608
```

Sample Output

For each test case, your program must output the value of c, followed by a space, followed by the color used to render that value of c according to the table above. The color should be printed in uppercase letters. Decimal values should be printed as they were received from the input.

```
1.1+2.0i RED
-0.7+0.2i BLACK
-0.5+0.65i ORANGE
-0.5+0.608i BLUE
```