**Darts**

In Practice Mode

Introduction

Write a function that returns the earned points in a single toss of a Darts game.

[Darts](https://en.wikipedia.org/wiki/Darts) is a game where players throw darts to a [target](https://en.wikipedia.org/wiki/Darts#/media/File:Darts_in_a_dartboard.jpg).

In our particular instance of the game, the target rewards with 4 different amounts of points, depending on where the dart lands:

* If the dart lands outside the target, player earns no points (0 points).
* If the dart lands in the outer circle of the target, player earns 1 point.
* If the dart lands in the middle circle of the target, player earns 5 points.
* If the dart lands in the inner circle of the target, player earns 10 points.

The outer circle has a radius of 10 units (This is equivalent to the total radius for the entire target), the middle circle a radius of 5 units, and the inner circle a radius of 1. Of course, they are all centered to the same point (That is, the circles are [concentric](http://mathworld.wolfram.com/ConcentricCircles.html)) defined by the coordinates (0, 0).

Write a function that given a point in the target (defined by its real cartesian coordinates x and y), returns the correct amount earned by a dart landing in that point.

Getting Started

Make sure you have read the "Guides" section of the [C track](https://exercism.io/my/tracks/c) on the Exercism site. This covers the basic information on setting up the development environment expected by the exercises.

Passing the Tests

Get the first test compiling, linking and passing by following the [three rules of test-driven development](http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd).

The included makefile can be used to create and run the tests using the test task.

make test

Create just the functions you need to satisfy any compiler errors and get the test to fail. Then write just enough code to get the test to pass. Once you've done that, move onto the next test.

As you progress through the tests, take the time to refactor your implementation for readability and expressiveness and then go on to the next test.

Try to use standard C99 facilities in preference to writing your own low-level algorithms or facilities by hand