COP 2930 - Pair Programming Assignment #1

Due date: Please consult WebCourses for your due date/time

Directions: Please pick one person in the class to work with and sign up as a group in the group set PairProg1. This process will be covered in class.

Objectives

1. To give students practice working together to solve a problem.
2. To learn how to apply complex if statements to solve a problem with several cases.
3. To learn how to use loops to solve a problem that may have many steps.
4. To practice creativity via the Python Turtle.

Problem A: Biking (biking.py)

You notice that after biking a while, you tend to slow down, because you've gotten tired. In particular, for the first 30 minutes your speed is 28 miles/hour, but the next thirty minutes your speed is 20 miles/hour, and for the third 30 minutes your speed is 14 miles per hour, and you have discovered that after 90 minutes, your speed drops to 10 miles per hour, but miraculously, you can continue biking at this speed forever!

Write a program that asks the user how long, in minutes, they want to ride their bike and prints out how far they'll bike in that time, in miles. Note that the answer may not be a whole number, so print a decimal.

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

The number of minutes the user bikes will be a positive integer less than or equal to 1440. (This is 24 hours!!!)

Output Specification

Produce a single line of output with the following format:

```
You will bike a distance of D miles.
```

where D your distance in miles you've biked over the input time.

Output Sample

Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

```
How many minutes did you bike?
40
You will bike a distance of 17.333333333333332 miles.
```

Sample Run #2

```
How many minutes did you bike?
90
You will bike a distance of 31.0 miles.
```

Sample Run #3

```
How many minutes did you bike?
200
You will bike a distance of 49.33333333333333 miles.
```

Note: Your answer doesn't need to match exactly to what is shown above, but it should be extremely close, numerically.

Problem B: Where Am I? (where.py)

Imagine that you are traveling on the Cartesian grid, starting at (0, 0) and want to arrive at (x, y), where x and y are specified by the user. On each move, the user can choose to move in the x direction or the y direction, and can choose how many units to move. Your program should prompt the user to keep on entering their moves until they arrive at the target destination. When they get there, your program should print a message stating how many moves it took them to arrive at their destination.

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

Both x and y will be integers in between -100 and 100, inclusive.

When user's enter which direction they want to move it, they will always enter either the character 'x' or the character 'y'.

The amount they want to move in their specified direction will be an integer in between -100 and 100, inclusive.

The user will always enter numbers such that she arrives at her destination in no more than 10 moves.

Output Specification

After each move, print a statement with the following format.

```
You are now at (x, y).
```

where (x, y) specifies the grid coordinate the user is currently at.

When the user reaches the destination, print out a message with the following format:

```
Congrats, you got to your destination in M moves.
```

where M is the number of moves it took to get to the destination.

Output Sample

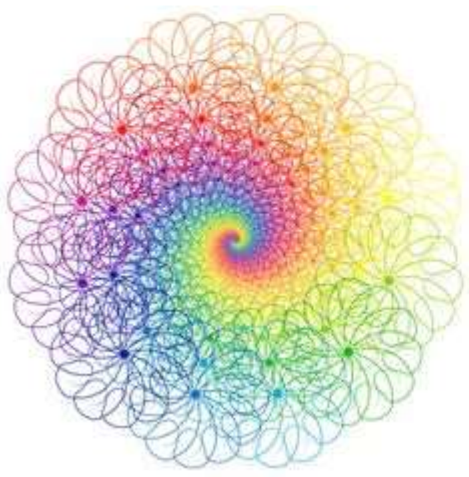
Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

```
You are currently at (0,0).
What is the x coordinate of your destination?
12
What is the y coordinate of your destination?
-33
What direction do you want to move(x/y)?
x
How much do you want to move in that direction?
-3
You are now at (-3, 0).
What direction do you want to move(x/y)?
y
How much do you want to move in that direction?
-20
You are now at (-3, -20).
What direction do you want to move(x/y)?
x
How much do you want to move in that direction?
15
You are now at (12, -20).
What direction do you want to move(x/y)?
y
How much do you want to move in that direction?
-13
You are now at (12, -33).
Congrats, you got to your destination in 4 moves.
```

Problem C: Turtle Design

The turtle can draw lots of cool things, especially if you use loops! Create a turtle drawing that looks like some sort of spirograph, using loops. I don't expect yours to be this ornate, but here is an example of a spirograph:



Restrictions

Please IDLE 3.6 (or higher) to develop your program. Write each in a separate file with the names specified previously, **biking.py** and **where.py**. You can call your third program whatever you want (except you can't call it turtle.py!!!)

Each of your team's **three** programs should include a header comment with the following information: both group members' names, course number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility to IDLE.