CS2250 Spring 2018 Hauschild

PROGRAMMING ASSIGNMENT 4:
Due date: 4/18/2018

Points: 100

In this assignment, you will be performing a horse racing simulation. To start with, you will write a Horse class. The Horse class should have the following member variables:

name: A string holding the horses name.
rider: A string holding the rider's name.
maxRunningDistPerSecond: An int that holds the maximum distance the horse could run in one second. This is a bound, so if it is set to 30, then the horse, every second, could go from 1 to 30 meters in a second. (Yes, a crude way to try and represent that horses vary in speed)
distanceTraveled: How far the horse has gone already.
racesWon: An int that determines how much races this horse and rider have won

In addition, the class should have the following constructors and member functions:

Constructor: This constructor should accept the horse's name and rider as arguments. It should initialize each horse to a random maxRunningDistPerSecond(1-100). DistanceTraveled should be set to 0.

Accessors and mutators: Appropriate accessor and mutator functions should be used as required by the following methods.

runASecond: A method that adds to distanceTraveled an amount from 0-maxRunningDistPerSecond. This is the method that moves this horse some distance towards the end of the race.

sendToGate: Reset the horse to the start of a race by setting distanceTraveled to 0.

toString(int goalLength): This method should attempt to draw where the horse is along its race. Should try to graphically display in ascii how far the horse has gone on its way to goalLength. This should be scaled so it doesn't overrun the screen.

Some examples:
```
|--------------->              | Pharaoh, ridden by Mark
|--------------------->        | Secretariat, ridden by George
|------------>                 | Calamity Jane, ridden by Mary
```
So if a horse has "distanceTraveled" of 50 and the goalLength passed is 100, then the > for the horse should be halfway towards the "goal". If it is called where the distanceTraveled is greater than the goalLength (indicating it has finished), this output would be:
```
|-------------------------------|>John
```
indicating the horse has finished the race.

Demonstrate this class in a program by first prompting the user for the number of horses in the race. Then create a vector of Horse objects, filling it up as appropriate. Then prompt the user for a distance to race. Since horses can travel in any time interval a max of 100, a good distance of 1000 could be appropriate to test.

In a loop, start the race. In this loop, for each interval of the loop, execute runASecond for each horse. Then output all the horses toString to show the user the current race status. Then prompt the user if they want to continue the race. Keep doing this until one of the horses has "won" the race. If there is a tie, break the tie by distanceTraveled. If there is a tie in distanceTraveled, you can choose any way you want to handle this tie.

Your program should allow multiple races with the same horses. At the end of each race, display the winning horse and statistics on how many each horse has won or lost.

You can include extra methods or attributes if you want, but make sure to only make members public if it is absolutely necessary.

This project will require 3 files, with your executable based on your name:
<lastnamefirstname>p3.cpp, the main driver program.
Horse.h: The header file for the Horse class
Horse.cpp: The file defining the Horse methods

Your output should look very similar to the following:

How many horses are in the race: 3
Please give me the name of horse 1: Pharaoh
Please give me the rider of rider 1: Mark
Please give me the name of horse 2: Secretariat
Please give me the name of rider 2: George
Please give me the name of horse 3: Calamity Jane
Please give me the name of rider 3: Mary
Please enter the distance of the race: 500
The start!
|> | Pharaoh, ridden by Mark
|> | Secretariat, ridden by George
|> | Calamity Jane, ridden by Mary

Are you ready for the next second(y/n)?:y
| > | Pharaoh, ridden by Mark
| > | Secretariat, ridden by George
| > | Calamity Jane, ridden by Mary
Are you ready for the next second(y/n)?:y
| > | Pharaoh, ridden by Mark
| > | Secretariat, ridden by George
| > | Calamity Jane, ridden by Mary
and so on, until a winner is found, which would end with some output like this:
| |> Pharaoh, ridden by Mark
| > | Secretariat, ridden by George
| > | Calamity Jane, ridden by Mary

Pharaoh has won 1/1 races.
Secretariat has won 0/1 races.
Calamity Jane has won 0/1 races.
Do you wish to continue(y/n)?:

Hint: Do this incrementally! First set up the "shell" with a very basic Horse class with a constructor and one attribute. With the 3 files, then make sure in main() that you can create a Horse object and then manipulate it. Do this BEFORE DOING ANYTHING ELSE. Do not try and implement the game and then later on add classes to it. Once you have the syntax of setting up a compilation of 3 files, then start adding features.