It's all a function of horsepower (A3)

89/100 Points

1/27/2023

Attempt 1

Review Feedback
1/27/2023

Attempt 1 Score:
89/100

View Feedback

Anonymous Grading: **no**

Unlimited Attempts Allowed

Details

The Project

Build a program that simulates a horse race. You will have a track of 15 units long, and five horses. On each turn, each horse will 'flip a coin' (Right, I don't know how horses flip coins either, but go with me on this.) Half the time, the horse will move forward one unit, and half the time it will stay in the same place. On each turn, you will see the track with the positions of the five horses, and dots for all the other positions. The race ends when one of the horses passes the finish line.

In this case we will do automated testing against your program. In order to do this you will prompt (ask for input) for a random seed and use it to seed your random number generator. This will allow you to test your program against several seeds. And we will test your program with a series of these random seeds. And this will not be your entire grade, but your ability to pass these tests is a part of your grade. This is a common industry practice and one we will try to introduce you.

Sample Run

You can find an HTML5 version here:

http://cs.iupui.edu/~ajharris/240/horseRace.html **(http://cs.iupui.edu/~ajharris/240/horseRace.html)**

Note that you will not need the automation feature nor a GUI. This just gives you an idea of the race concept.

Here is sample output of the program (Your output should match this format exactly for the best possible grade, but if you wish to add prompts for an enter key between rounds, you will not be penalized.):

```
Please enter a random seed: 700
.0.............
.1.............
.2.............
3..............
4..............


..0............
..1............
..2............
.3.............
.4.............


...0...........
..1............
...2...........
.3.............
```

```
.4.....


...0.....
..1.....
....2.....
.3.....
.4.....


...0.....
..1.....
....2.....
..3.....
..4.....


...0.....
...1.....
....2.....
...3.....
...4.....


...0.....
...1.....
....2.....
....3.....
...4.....


...0.....
...1.....
.....2.....
.....3.....
...4.....


...0.....
...1.....
.....2.....
......3.....
....4.....


...0.....
...1.....
......2.....
.......3.....
....4.....


....0.....
...1.....
.......2.....
........3.....
.....4.....


....0.....
...1.....
.......2.....
........3.....
......4.....


.....0.....
...1.....
.......2.....
........3.....
......4.....


.....0.....
....1.....
.......2.....
........3.....
......4.....
```

```
.....0.........
....1..........
........2......
..........3....
......4........


.....0.........
.....1.........
........2......
...........3...
.......4.......


.....0.........
......1........
.........2.....
............3..
.......4.......


.....0.........
.......1.......
.........2.....
............3..
.......4.......


.....0.........
........1......
.........2.....
............3..
.......4.......


.....0.........
........1......
..........2....
............3.
.......4.......


......0........
.........1.....
...........2...
............3.
.......4.......


.......0.......
.........1.....
...........2...
..............3
.......4.......


........0......
..........1....
............2..
..............3
........4......


.........0.....
..........1....
............2..
..............3
........4......


.........0.....
..........1....
............2.
..............
........4......

Horse 3 wins!
```

Algorithm

As always, begin with an algorithm in markdown this time. This document should begin by describing the main purpose of the program using the Goals - Input - Output - Steps technique we've been using. For each function you have described in the main program, you must go through the same process. Remember that your algorithm should be complete enough that by the time you are ready to start writing code, you pretty much know what you are going to do. You should get mostly through the algorithm step together in the recitation center, but definitely try to write this on your own as well. Your algorithm file should be named algorithm.md.

The code

Once you have a solid algorithm plan, please write the code in C++. It should be a reasonably straightforward process to convert each step of your algorithm to a line or two of C++ code. It's absolutely likely that you will have to look up a few concepts. Your code should compile with g++ on tesla. If you know how to use make and git, these are extremely helpful tools.

Tips and Strategies

- **Central data structure** - The key to this program is determining a central data structure that contains the main information you'll need. Generally beginners make this much too complicated. I think an array of five integers is the cleanest solution. Think about this before you do anything else.
- **Use functions** - This code is too long to be done in a main function. You'll need to determine which functions you need and what they should do.
- **No objects yet** - We will re-write this program using object-oriented syntax in the next exercise. For now, you will write it in a procedural style. You can use any pre-build objects in base C++, but don't build custom objects yet. We will compile with C++ 98 by default, so no vectors etc. This is however, something that you can explore as a potential blackbelt is implementing it in a more recent c++ version and explaining the differences.
- **Random numbers** - You will need a random number. You might need to look up how this is done in C++. it is *not* the same way you might have done in another language.
- **Random seeding** If you always get the same response, you may need to look into something called 'seeding the random number generator.' It is not difficult, but you need to do it to get truly unique results.
- **string manipulation** - It seems obvious that the tracks are strings, but that may not be the easiest way to work with them. It may be much easier to simply print out a bunch of characters based on the horse's position.
- **Globals** You may use global constants (My solution used two.) No other global variables are required, but for simplicity you may create a global array for the main data element
- **Pausing each turn** - the sample code shows the complete run of the game, but if you wish, you can pause each turn and ask the user to press enter for the next turn. That will make the output slightly different than the example, but more fun for the user. You can use std::cin.ignore() to wait for the next enter key.

Turning in the project

- This assignment will be turned in through IU GitHub. Please name your repo CSCI24000_spring23_A3
- If your repo has another name, it will not be graded.
- *After you've submitted your assignment in Github, come back to this assignment page and submit the full Github URL (**https://github.iu.edu/username/reponame** (**https://github.iu.edu/username/reponame**)) for your repo here.*

- Your repo must be private. We will not grade projects in public repos, and we will require you to take down any homework assignments in a public repo.
- We will expect your base assignment to be in a folder called base and any blackbelt extension to be placed in a folder called blackbelt (if applicable), with instructions and a description of what you did.
- Your source code is expected to be in a file called horse.cpp, failure to follow this may result in your code not being found by the grader or our automated testing (if any).

Please ensure to add all the following as collaborators:

- **Git Collaborators (https://iu.instructure.com/courses/2131288/pages/assignments-submission-and-grading)**

Black Belt Extension

In my algorithm, I treated the array of horses as a global variable. While this is a defensible position in this particular example (as every function refers to the array) it is not the best approach. Redesign this program so the array of horses is declared in the main function and passed as a parameter. Remember the relationship between arrays and pointers, as this will help.

---

https://github.iu.edu/parmsing/CSCI24000_spring23_A3

(https://iu.instructure.com/courses/2131288/modules/items/28790587)

(https://iu.instructure.com/courses/2131288/modules

You are unable to submit to this assignment as your enrollment in this course has been concluded.