**COS-241: DATA STRUCTURES**

**Problem 2: Simulation: The Tortoise and the Hare**

**DESCRIPTION**

This project deals with the classic race between the tortoise and the hare. For this project, you be using random-number generation to move the creatures. To make things more interesting, the animals have to race up the side of a slippery mountain, which could cause them to lose ground. In this race either animal could win or there could be a tie with no clear winner.

The animals begin at "Square 1" of 70 squares. Each of the squares represents a position the animal can hold along the racetrack. The finish line is at Square 70. When an animal wins, a winning message of your choice should be posted. For example:

* Yay! The rabbit won! He hops the fastest!
* Woo-hooo! Slow and steady wins the race! Congratulations, turtle!
* Tie score—no winner! Want to race again?

To start the race, print a message similar to:

* Bang! Off they go!

There is a clock that ticks once per second. With each tick of the clock, your program should adjust the position of the animals according to the following rules:

|  |  |  |  |
| --- | --- | --- | --- |
| **Animal** | **Move Type** | **Percentage of the Time** | **Actual Move** |
| Tortoise | Fast Plod | 50% | 3 squares to the right |
| Tortoise | Slip | 20% | 6 squares to the left |
| Tortoise | Slow Plod | 30% | 1 squares to the right |
| Hare | Sleep | 20% | No move at all |
| Hare | Big Hop | 20% | 9 squares to the right |
| Hare | Big Slip | 10% | 12 squares to the left |
| Hare | Small Hop | 30% | 1 square to the right |
| Hare | Small Slip | 20% | 2 squares to the left |

Keep track of the positions of the animals by using variables. If an animal slips, the lowest it can go is back to position 1. The highest it can go is to position 70 when it wins the race.

You will work with the percentages in the table above by generating a random integer **current** in the range

**1≤ current ≤10**.

For the tortoise, a "fast plod" is when **1≤ current ≤ 5**, a "slip" when **6 ≤ current ≤ 7**, or a "slow plod" **8 ≤ current ≤ 10**. A similar approach would be used to set up the moves for the hare.

For each tick of the clock (each repetition of the loop), print a 70-position line showing the letter **T** in the tortoise’s position and the letter **H** in the hare’s position. If the two animals land on the same square, which may happen, the animals will bump. If this happens, print **BUMP!** at the current position.

After you print each line, check to see if either animal has landed on Square 70. If this happens, print a winning-type message.

It may make the simulation more interesting if you have users press any key after each iteration of the loop, so that they can see the movement of the animals.

**DELIVERABLES**

* Your C++ source code with any header files
* Your executable code
* A document detailing how you will test your program that also includes screenshots of a few sample runs of your program.
* An overview document giving the name of each file submitted and its purpose, as well as any discussion you have on implementing this program. If there are any problems with your  program or it does not run or run as it is supposed to, please indicate that as well in this document.