Tutorial Problem Set #6

Due: Wednesday, October 30, 2024, 11:59 PM

Policy

- Piazza questions on tutorial problems will be ignored or deleted. Questions will only be answered in your assigned tutorial section.
- Sample executables can be found in your 1249 git repository directory (run `git pull`).
- Completing the problem set will reduce the weight of the final exam by 0.5%. To complete a problem set, you must pass at least 50% of the public and secret tests.
- You may assume all input is valid. Tutorial problems **NEVER** require checking for invalid inputs.

Question 1

The Tortoise and the Hare is a famous story of a race between a slow-moving tortoise (similar to a turtle) and a fast-moving hare (similar to a rabbit). The hare dashes out ahead of the tortoise, but then figures he has time to take a nap, and is overtaken by the tortoise while he is sleeping. You will now re-enact this race in C++.

Consider the following partial implementation of the base class `Racer`:

```
class Racer {
    unsigned int distance;
  public:
    Racer(); // initializes distance to 0
    unsigned getDistance() const;

    // declare/define any other methods you need here
};
```

a) Modify the `Racer` class above, and define two subclasses of class `Racer`, called `Hare` and `Tortoise`, with the following characteristics:

- both subclasses provide a method `tick`, which takes no arguments, but updates the `distance` field to reflect the passage of one unit of time
- in `Tortoise`, each unit of time (i.e., each call to `tick`) increases the `distance` field by 1.
- in `Hare`, each unit of time (i.e., each call to `tick`) increases the `distance` field by 2 until 10 units of time have passed (i.e. up to and including the 10th time `tick` is called), which is when the `Hare` takes a nap. Thereafter, a call to `tick` has no effect on the `distance` field until the 40th call to `tick` (counting from the beginning) when the `Hare` wakes from its nap. From the 41st call onward (and including the 41st call), a call to `tick` again increases the `distance` field by 2.
- every racer is either a `Hare` or a `Tortoise`; there are no generic racers.

b) Write a `main` function that declares two pointers of type `Racer*`, one pointing to a `Hare` object and one to a `Tortoise` object, and runs a race between them. The program will then read a positive integer from standard input that specifies the number of units, `N`, that the race will last. The race lasts until one of the racers has covered `N` units of distance. Your main program should run a loop that keeps calling the `tick` method of each `Racer`, stopping when one of them reaches `N` units of distance. Print out `"Tie"` if they both reach `N` units of distance in the same time; otherwise print out either `"Hare"` or `"Tortoise"` to indicate the winner.

Submission

Submit your `.cc` and `-impl.cc` files in a ZIP file called `tut06.zip` to Marmoset.