CIS2107_Lab07: "Race"

Points: **100** points

Objective:

- To design and implement **functions** to process pointers.

Instructions:

- Be sure to document your code (add comments on top of each function).
- In the comments add your name, date, course, homework number, and statement of problem.

Steps:

In this `Race.c`, you'll recreate one of the truly great moments in history, namely the classic race of the tortoise and the hare. You'll use random number generation to develop a simulation of this memorable event.

Our contenders begin the race at "square 1" of 70 squares. Each square represents a position along the racecourse. The finish line is at square 70. The first contender to reach or pass square 70 is rewarded with a pail of fresh carrots and lettuce. The course weaves its way up the side of a slippery mountain, so occasionally the contenders lose ground.

There's 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 rules:

Animal	Move Type	Percentage of the time	Actual Move
Tortoise	Fast plod	50%	3 squares to the right
	Slip	20%	6 squares to the left
	Slow plod	30%	1 square to the right
Hare	Sleep	20%	No move at all
	Big hop	20%	9 squares to the right
	Big slip	10%	12 squares to the left
	Small hop	30%	1 square to the right
	Small slip	20%	2 squares to the left

Functions Use variables to keep track of the positions of the animals (i.e., position numbers are 1–70). Start each animal at position 1 (i.e., the "starting gate"). If an animal slips left before square 1, move the animal back to square 1.

Generate the percentages in the preceding table by producing a random integer, i, in the range `1 ≤ i ≤ 10`.
For the tortoise, perform a "fast plod" when `1 ≤ i ≤ 5`, a "slip" when `6 ≤ i ≤ 7`, or a "slow plod" when `8 ≤ i ≤ 10`.
Use a similar technique to move the hare.

Begin the race by printing

```
BANG!!!!!
AND THEY'RE OFF!!!!!
```

Then, for each tick of the clock (i.e., each repetition of a loop), print a 70-position line showing the letter T in the position of the tortoise and the letter H in the position of the hare. Occasionally, the contenders will land on the same square. In this case, the tortoise bites the hare, and your program should print OUCH!!! Beginning at that position.

All print positions other than the `T`, the `H`, or the `OUCH!!!` (In case of a tie) should be blank.

After each line is printed, test whether either animal has reached or passed square 70. If so, then print the winner and terminate the simulation.

- If the tortoise wins, print `TORTOISE WINS!!! YAY!!!`
- If the hare wins, print Hare wins. `Yuch`.
- If both animals win on the same tick of the clock, you may want to favor the turtle (the "underdog"), or you may want to print It's a tie.
- If neither animal wins, perform the loop again to simulate the next tick of the clock.

Use function `sleep()`from `<unistd.h>` to slow down the game and feel the excitement!

Sample run:

```
ON YOUR MARK, GET SET
BANG               !!!!
AND THEY'RE OFF    !!!!
 H T
H   T
       T H
         HT
             T                                                       H
Hare wins. Yuch.
```

```
ON YOUR MARK, GET SET
BANG               !!!!
AND THEY'RE OFF    !!!!
   T     H
    T     H
          H                                                          T

TORTOISE WINS!!! YAY!!!
```

```
ON YOUR MARK, GET SET
BANG               !!!!
AND THEY'RE OFF    !!!!
   T     H
      T   H
       T   H
          TH
         H   T
          H     T
           H     T
                    OUCH!!!
                     OUCH!!!
                      H T
                       H T
           H              T
H                          T
                            T                                        H
Hare wins. Yuch.
```