COMP 2401B -- Assignment #1

Due: Thursday, October 4, 2018 at 12:00 pm (noon)

Goal

Using the VM provided for the course, you will implement a small C program to simulate a race between a tortoise and a hare. Timmy Tortoise and Harold the Hare are racing each other along a narrow path up the Mount of Doom. Runners take turns moving up the hill, and each runner exhibits randomized behaviour from a predetermined set of moves. These moves include climbing quickly or slowly, or slipping back down the hill, or even taking a nap. You will simulate the race and declare a winner when one of the runners reaches the top.

Base code to initialize the random number generator has been provided and posted in *cuLearn.*

Learning Outcomes

You will practice problem solving and designing modular functions to implement a solution. You will also work with arrays in C.

Instructions

1. Data structures

Your program will:

- represent the path up the Mount of Doom as an array of characters, up to a maximum position (`MAX_POS`) of 70
  - if a given position (index) in the array contains a runner, the value in the path array at that position will be the runner's avatar ('T' for Timmy and 'H' for Harold)
  - if a position has no runner, the path array will hold a space character
- store each runner's current position on the path, as an integer corresponding to the index in the path array

2. Program behaviour

You will design modular functions to implement the following functionality. Your program will:

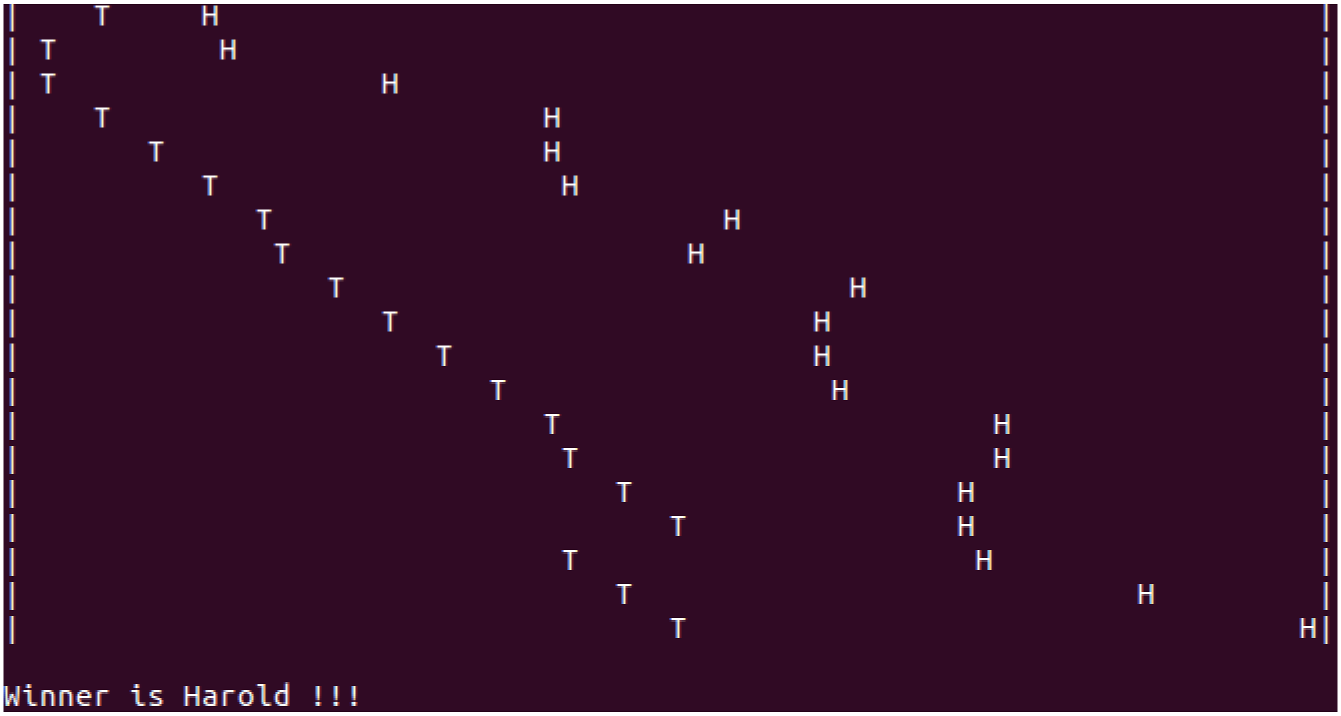
- loop until one of the runners reaches the top; in every iteration:
  - each runner makes a random move, based on the information in Table 1
  - the runner's position is updated, and his avatar on the path is moved
  - the path is printed to the screen
- once a runner reaches the top, print the name of the winner to the screen
- a sample program execution is shown in Figure 1; each line represents the path after one iteration, with the bottom of the hill on the left-hand side and the top of the hill on the right

3. Documentation

You will thoroughly document your program and every function, as discussed in class. Every function will indicate whether each parameter is an input parameter, an output parameter, or an input-output parameter.

Table 1 -- Runner Moves

Runner	Type of move	Random chance	What happens
Tortoise	Fast plod	50%	move 3 positions up
	Slow plod	30%	move 1 position up
	Slip	20%	move 6 positions down
Hare	Big hop	20%	move 9 positions up
	Small hop	30%	move 1 position up
	Big slip	10%	move 12 positions down
	Small slip	20%	move 2 positions down
	Sleep	20%	no move



```
|    T     H                                                             |
| T         H                                                            |
| T                  H                                                   |
|    T                        H                                          |
|       T                     H                                          |
|          T                   H                                         |
|             T                         H                                |
|              T                      H                                  |
|                 T                            H                         |
|                    T                       H                           |
|                       T                    H                           |
|                          T                  H                          |
|                             T                        H                 |
|                              T                       H                 |
|                                 T                  H                   |
|                                    T               H                   |
|                              T                      H                  |
|                                 T                            H         |
|                                    T                                  H|

Winner is Harold !!!
```

Figure 1 -- Sample execution

Constraints

- your program must be correctly designed and separated into modular, reusable functions
- your program must reuse functions everywhere possible
- your program must perform all basic error checking
- your program must be thoroughly documented
- do not use any global variables
- if skeleton code is provided, you must not make any changes to the existing code

Submission

You will submit in *cuLearn*, before the due date and time, one `tar` or `zip` file that includes the following:

- all source code, including the code provided, if applicable
- a readme file that includes:
  - a preamble (program and revision authors, purpose, list of source/header/data files)
  - the exact compilation command
  - launching and operating instructions

Grading (out of 100)

Marking components:

- 10 marks: correct data structure initialization
- 80 marks: correct racing behaviour
  - 8 marks: correct loop header
  - 30 marks: correct implementation of moving behaviour for the tortoise
  - 35 marks: correct implementation of moving behaviour for the hare
  - 7 marks: correct printing of the path at every iteration
- 10 marks: correct computation and declaration of winner

Deductions:

- Packaging errors:
  - 100 marks for an incorrect archive type that is not supported by the VM
  - 50 marks for an incorrect archive type that is supported by the VM
  - 10 marks for a missing readme
- Major programming and design errors:
  - 50% of a marking component that uses global variables
  - 50% of a marking component that is incorrectly designed
- Minor programming and design errors:
  - 10% for consistently missing comments or other bad style
  - 10% for consistently failing to perform basic error checking
- Execution errors:
  - 100% of a marking component that cannot be tested because it doesn’t compile or execute in VM
  - 100% of a marking component that cannot be tested because it’s not used in the code
  - 100% of a marking component that cannot be proven to run successfully due to missing output