CSC 362 Programming Assignment #3 Due Date: Thursday, October 16

You are probably familiar with the story of the hare and the tortoise. The hare, being much faster, should win the race. The hare is over confident and takes a nap. While sleeping, the tortoise overtakes him. In this assignment, you are to write a program that will simulate a race between the hare and tortoise. The primary purpose of this assignment is to use pointers that point to the characters of a C array (string).

The race will take place on a narrow path. The path consists of open spaces, rocks (which the hare must hop over, but the tortoise can crawl over), a couple of streams (again, the hare must hop over each stream, the tortoise can swim through the stream), and a carrot patch (if the hare lands in the carrot patch, it will eat a carrot instead of taking its turn). The hare hops up to 8 paces at a time (randomly generated) but cannot hop *onto* a rock or hop *into* a stream. If the hare is to move onto a rock or into a stream on its next turn, it must instead stop right before the rock or stream. If the hare lands in the carrot patch, during its next turn, it will eat the carrot. In the next turn, there is no carrot at its current position so it can move but if it lands in a spot with another carrot, it again waits a turn to eat the carrot. To keep track of this, your code will have to remove any carrot that has been eaten from the path. Additionally, there is a 50/50 chance that the hare will take a turn off to nap. The tortoise crawls up to 3 paces at a time (randomly generated), but any single turn could leave the tortoise on a rock or in a stream or in the carrot patch. Finally, the hare and tortoise cannot occupy the same spot (other than at the start of the race), so if one lands on the other, it has to move back to the space immediately beforehand> But if this is the hare, and moving backward places it in a stream or on a rock, it must move back to the square before the stream or rock. Your program should work as follows:

1) Set up the path (see page 2), which will be stored as an array of characters
2) Set up two char pointers to point to where the hare and tortoise currently are in the path respectively, initially pointing them at path[0]
3) Do for each turn until one animal wins
    a. Tortoise's turn: generate a random move of 1-3 paces, if the tortoise lands on the hare, then the tortoise must stop 1 space short of the hare, move the tortoise pointer to the new location
    b. Hare's turn: if the hare is in the carrot patch on the location of a carrot, the hare eats the carrot instead of moving (replacing the 'C' in the path to ' '); otherwise, generate a random number to determine if the hare takes a nap (50/50 chance). If the hare does not nap, generate a random move of 1-8 paces to move the hare. If the move takes the hare onto a rock, into a stream, or onto the tortoise, it must stop short by backing up into the first available free space before the obstacle (note that there may be multiple obstacles in a row, for instance the tortoise might be on the spot immediately before a rock or a stream). Move the hare pointer to the new location. If the hare naps, do nothing.
    c. Print out the current path, including the locations of the hare and tortoise (print a 'T' for tortoise, 'H' for hare, 'S' for stream, 'R' for rock, 'C' for carrot, and ' ' for an open space). If the tortoise is on a rock, in a stream or at a carrot, output 'T', instead of 'S', 'R' or 'C'. If a collision occurred during this turn between the two animals, output the word collision after you output the path. Also output the turn number at the end of this line.
4) Output the winner (whichever animal has reached the furthest; note that both may reach the finish line in the same turn but one will have gone farther).

Note: the end of the race is at path+50 so your while loop (3) should iterate while h and t are < path + 50. Also, because in moveHare you will have to test hare to see if it has landed on a 'R', 'S' or 'C', you will do *hare. But if hare >= path+50, *hare will attempt to dereference beyond the path array which is a memory violation runtime error. So whenever you use *hare, make sure hare < path+50.

The path is stored in a string (array of 50 characters), comprised of the characters ' ', 'R', 'S', and 'C'. You may set up your path array using normal array notation, but from there on, accessing into the array **must** be done through pointers and pointer arithmetic. Aside from main, write 3 (or more) functions: one to move the tortoise, one to move the hare, one to print out the path. You will have to pass pointers to these functions.

The function to move the tortoise must know whether the tortoise collides with the hare. The function to move the hare must know if the hare is currently on a carrot, or whether the move causes the hare to collide with the tortoise or land the hare on a rock or in a stream (so you will have to pass it the path and the tortoise pointer). The print function needs all 3 pointers (path, tortoise, hare). You MUST use pointer arithmetic to move the hare and tortoise pointers. That is, you will be adjusting the values stored in *pointers* to move the animals. Do not just store the hare's location or tortoise's location as an int value between 0 and 49. Each of these move functions should return the pointer manipulated. For instance, you might call the function using code like `hare=moveHare(hare, tortoise, path);` As an alternative, you could do `moveHare(&hare, tortoise, path);` but this would be more complicated and is not recommended. In the print function, use a for-loop or while loop to iterate down the path array (using a pointer to point at each character), ending when the pointer points to \0. Compare the pointer to each animal pointer to see if they equal. If so, print out the proper letter for the animal ('T' or 'H'), otherwise print the character at the given position in the array (' ', 'R', 'S', 'C'). Also output whether a collision occurred during the turn. As collisions are determined in the moveTortoise and moveHare functions, you will need an additional variable passed as a parameter to these two functions and then to the print function. Pass this parameter appropriately as it may change values in one of the move functions. During each iteration of the main loop (looping until someone wins the race), count the turn number and include this in the output. Outside of the loop, output who won the race.

While you need (a minimum of) 3 functions, you can place them all in a single file. If you do so, you do not need a separate header file.

The path is 50 paces long, and consists of open space except as follows:

Rocks at locations 2, 4, 7, 17, 19, 23, 27, 42, 48
Streams at 9-13 and 45-47.
The carrot patch is from 30-39

Run your program as many times as necessary to obtain one win for the Hare and one win for the Tortoise. Roughly half the time, the Tortoise should win. If you have to run your program a lot of times (say more than 10 or 20) before getting both animals to win, then you probably have a logical error somewhere.

Hand in your source code and two outputs (one for each animal winning). A sample output is given below. Remember: H is for Hare, T for Tortoise, R for rock, S for stream and C for carrot. Notice as the hare moves through the carrot patch, it eats carrots, leaving blanks behind in the output.

```
Turn:    1     RTRH R SSSSS   R R   R   R  CCCCCCCCCC  R  SSSR
Turn:    2     R T  RHSSSSS   R R   R   R  CCCCCCCCCC  R  SSSR
Turn:    3     R R  THSSSSS   R R   R   R  CCCCCCCCCC  R  SSSR
Turn:    4     R R  RHTSSSS   R R   R   R  CCCCCCCCCC  R  SSSR  --Collision--
Turn:    5     R R  R SSSTSH  R R   R   R  CCCCCCCCCC  R  SSSR
Turn:    6     R R  R SSSSTH  R R   R   R  CCCCCCCCCC  R  SSSR  --Collision--
Turn:    7     R R  R SSSSSHT R R   R   R  CCCCCCCCCC  R  SSSR  --Collision--
Turn:    8     R R  R SSSSSH TR R   R   R  CCCCCCCCCC  R  SSSR
Turn:    9     R R  R SSSSS   R T H R   R  CCCCCCCCCC  R  SSSR
Turn:   10     R R  R SSSSS   R RT HR   R  CCCCCCCCCC  R  SSSR
Turn:   11     R R  R SSSSS   R R T R   RH CCCCCCCCCC  R  SSSR
Turn:   12     R R  R SSSSS   R R   RT  R  CCCHCCCCCC  R  SSSR
Turn:   13     R R  R SSSSS   R R   R   T  CCC CCHCCC  R  SSSR
Turn:   14     R R  R SSSSS   R R   R   R  TCCCCC CCC  RH SSSR
Turn:   15     R R  R SSSSS   R R   R   R  CTCCCC CCC  R  SSSRH
Turn:   16     R R  R SSSSS   R R   R   R  CCCCTC CCC  R  SSSRH
Turn:   17     R R  R SSSSS   R R   R   R  CCCCCCTCCC  R  SSSR
Hare wins!
```