CS 410: Homework 3

Due: Tuesday, Oct. 2, 2018 at noon **100 pts**

Instructions: Create a subdirectory named "hw3" in your cs410 directory. Use that subdirectory for your all file submissions on this assignment. At the end of the homework assignment, these two files should be found in your hw3 directory:

1. a single C++ compilable file containing a program written in C++ named "hw3.cpp"
2. a "typescript" file demonstating program compilation, execution and testing. Use the commands below at the UNIX prompt to generate the typescript file:
    - **script** command to start a typescript.
    - **ls -l** to list files in your directory and write date/time
    - **cat hw3.cpp** to print out solution file
    - **g++ -o hw3 hw3.cpp** to compile program
    - **./hw3** to execute program
    - **exit** command to end typescript file

Background: Poor Hans. For most of us, life is a trial-live one day at a time and we usually do pretty well with minimal effort. However, for Hans Moleman, life is really tough. Being nearly blind, he has a hard time seeing, something most of us take for granted. Nevertheless, he got new lenses for his glasses and that should have solved his problem. But, alas, he got the lenses at the new super store in Springfield, Gal-Mart, which employs a Dr. Riviera who uses some ridiculous, crazy new formula for computing lens thickness. Now he can hardly see what he's doing or where he's going with his glasses on. SO, he needs to get back to Dr. Nick's to have his sight re-evaluated and new lenses made. The office is 1414 feet from his house. He cannot drive in his current condition, and he's scared to death of buses, so he's going to walk. The thing is, Hans may or may not make it to his intended destination!

Three possibilities exist:

1) he makes it to the doctor's office;
2) he ends up back home; and
3) after a long time, he's so far away that he is picked up by the local police and taken to a homeless shelter.

You're going to write a program that will simulate his walk across town. In fact, your program will generate 250 possible walks and gather statistics on the results.

Specifications: Your program will simulate 250 walks. A "walk" begins at home and proceeds from decision point to decision point until the walk terminates. Termination happens when a decision point is:

- within 500 feet of Dr. Nick's office
- within 50 feet of home **and** there are at least 5 other decision points preceding this point

- the 400th decision point

The first of the above conditions implies that he is close enough to his intended destination to be able to see that he has arrived. The second implies that he has been walking for a while and hasn't actually gone anywhere, so he just gives up and goes back inside and watches re-runs of the Simpsons! DOH! The third simulates a lost old man wandering off into the horizon never to return.

Details: At each decision point, he will almost always turn right. This is because his eye-sight is so screwed up that his vision "pulls" him in that direction. However, every 7th decision point he turns left. Also, for the first walk, he will walk 20 feet away from his house to his first decision point. On the second walk, he walks 21 ft before the first decision point (note this is NOT the second step of the first walk), etc...Each **first** decision point is 1 foot further from the house than the previous one. You can choose the direction of this first move, but be consistent from walk to walk, starting in the same direction. And, the distance he walks after each decision point is given by:

$$d_0 = \textit{initial number of feet from the house}\left(20 \textit{ for } 1^{st} \textit{ walk},\ 21 \textit{ for } 2^{nd} \textit{ walk},\ \textit{etc}\right)$$

$$d_{i+1} = \left(\left[\left(d_i \times 124985 + 1367892\right) \bmod 1654872235\right] \bmod 300\right) + 21, i = 1,\ 2,\ 3,\ \ldots$$

If you are unfamiliar with an iterative formula like this, this is how it works. The first distance walked (as stated above) is 20 ft. You'll substitute this into the second equation above to compute the 2nd distance, d_2. Then substitute d_2 into the equation to compute d_3. Continue in this way. For his second walk, you'd start with first distance walked being 21.

The output from your program should be a listing of the instances when he either goes nowhere (ends up at home) or finds his way to Nick's office, and how many feet he traveled to get either of those two places. And, if the distance walked is greater than a mile, output that distance in miles. After that, output the percentages of times he got to Nick's, ended up at home, and got lost in the ozone.

Part of my output looks like this:

Ended walk at	**total dist traveled (ft)**
Nick's	17709 (~ 3.35398 miles)
home	9493 (~ 1.79792 miles)
Nick's	17663 (~ 3.34527 miles)
home	7803 (~ 1.47784 miles)
home	2161
home	2742
Nick's	18774 (~ 3.55568 miles)
home	9804 (~ 1.85682 miles)
home	15501 (~ 2.9358 miles)
Nick's	11444 (~ 2.16742 miles)
Nick's	17673 (~ 3.34716 miles)

Also: In order to code this problem, I'd suggest you establish a coordinate system for the location of Hans, the doctor's office and Hans's home. Since Hans starts his walks at home, put home at (0, 0). Since Nick's office is 1414 feet (straight line) away, put it at (1000, 1000). You can work out the rest from there. In order to compute the distance from one point to another, you will need to use the sqrt() function. It is contained in the library <cmath>. Just #include it like you do the iostream library. This function will return a float type and you will need to put what you wish to find the square root of inside the parentheses.

When you submit: Just submit, since there is no input from the user.

And, as always, let your TAs or instructor know if you need any help.