CS 103 Lab - ZOMBIE-POCALYPSE

1 Introduction

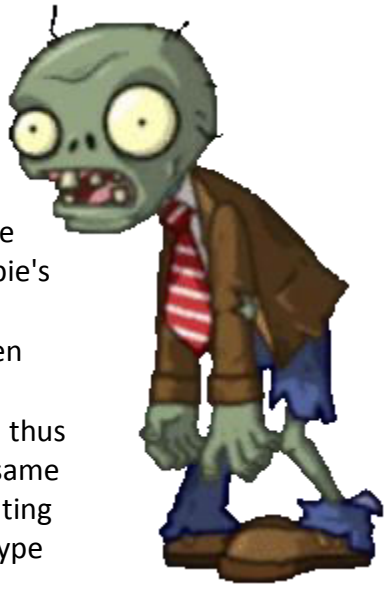
In this assignment you will use a computer program to help investigate the pressing question: How long might I survive once the zombie-pocalypse starts? This program will require you to utilize the following knowledge and skills:

- for and while loops
- conditionals
- variables
- input and output
- basic arrays
- simple algorithms

2 The Problem

In a population of N people of which k are initially zombies, how many nights are required before the entire population (all N people) become zombies if the following conditions hold:

- Zombies only come out at night
- Being bitten by a zombie turns a human into a zombie
- Each zombie randomly chooses one person (zombie or human) from the population per night and bites them. (A zombie may bite another zombie but this has no effect, other than wasting that zombie's bite for that night).
- On any given night, a zombie (being dumb) may even choose to bite itself which has no effect.
- A human who is bitten during a particular night and thus becomes a zombie should not bite anyone on that same night. They must wait until the next night to start biting people.Manipulate C-string variables (variables of type `char*`)



3 Setting up the Solution

While we can attempt to solve this problem analytically, let us instead use the computer to "simulate" the problem to find the answer. Suppose we have an array, pop, of length N that stores bool's representing the population, where the i-th entry represents person i from the population. pop[i] will equal true if that

person IS a zombie and false, otherwise. The first k entries in the array will be zombies while the remaining N-k will be human. For each zombie we can randomly choose someone to bite (who may or may not already be a zombie), changing them to a zombie if they were human. This would represent one night. We could repeat this process on a second night, third night, fourth night, and so on, counting the number of nights it takes until the entire population is a zombie.

By using the computers random number generator we can simulate zombie bites and count how many nights it takes until the population is all zombies. However, the answer we obtain would be largely influenced by the random number generation, since the numbers we generate will determine if a zombie bites a human or another zombie (and thus another human is safe). We can easily see that if the zombies get "lucky" (from the zombie's perspective), it may take a small number of nights to bite all remaining humans while an "unlucky" sequence of random values may mean a large number of nights before the population is all zombies, since they end up biting each other more often. To account for this issue, we will use the law of large numbers to perform not just 1 simulation of the zombi-pocalypse, but MANY (say, M). We can then use the average number of nights taken for each of the M simulations of the zombie-pocalypse to arrive at our answer. In addition, it may be interesting to find the range of answers over the M simulations, meaning what was the smallest (minimum) number of nights any zombie-pocalypse simulation required and what was the largest (maximum) number of nights any zombie-pocalypse simulation required. You should track these values as you perform the simulations and output them with the average.

To control the repeatability of our random simulations we will allow a seed value to be provided by the user. This seed value will be supplied to the random number generator so that execution of the program multiple times using the same seed will result in the same sequence of random numbers (and thus same answer).

4 Input/Output

Based on the description above your program should take in the following input values (in the order shown) from the user. The inputs may be typed on one line and separated by whitespace.

- N, the total population
- k, the number of initial zombies (Exit the program if the user enters k > N)
- M, the number of simulations of the zombie-pocalypse to run and average the results
- s, the seed for the random number generator

Your output should be the average, maximum and minimum number of nights until the entire population is a zombie. It should be formatted as:

```
Average Nights: avg-nights
Max Nights: max-nights
Min Nights: min-nights
```

Here are three sample runs of the program: the emphasized text is the part that the user typed in and the $ is the command-line prompt.

```
$ ./zombies
Enter the following: N k M seed
20 2 10000 137957
Average Nights: 7.3054
Max Nights: 16
Min Nights: 4
$ ./zombies
Enter the following: N k M seed
100 5 10000 137957
Average Nights: 10.0351
Max Nights: 19
Min Nights: 7
$ ./zombies
Enter the following: N k M seed
100 10 10000 371
Average Nights: 8.9769
Max Nights: 18
Min Nights: 6
```

5 Skeleton:

We have provided a skeleton (starter code) in Vocareum.

You'll notice we declare an array pop to represent the population. In a normal program, the size of this array would match the population size entered by the user. However, since statically declared arrays must be of a constant size (known at compile time), we allocate a LARGE array that will be big enough to handle the large population we will test your code with. In your program you should only use the first N locations in that array.

6 Readme:

Answer the experimental questions that are listed in the readme file.
Open “readme.txt” in Vocareum to read the questions and answer them.
Please write your answers in the same file “readme.txt”

7 Style:

You must follow all of the code style guidelines posted here.
http://bytes.usc.edu/cs103/coursework/style/

8 Submit:

Use Vocareum to submit your code and readme file. It will run some basic tests to check your formatting. To check it more exhaustively, it would be a good idea for you to try running it on additional test cases.

9 Q&A:

Q: Which k people should start as zombies?
A: Your code should work in a way where it does not matter. However, it is likely easiest to let the first k entries in the people array be the zombies.

Q: Can we use functions for this assignment?
A: You can if you like, but they are not required, and don't help in any serious way.

Q: What if the user enters 0, a negative number, a decimal, or text for N, k, M, or seed?
A: For this assignment, assume the user only enters positive integers.

Q: How do you compute minimums and maximums?
A: Section 4.7.4 has a closely related example. Store some memory, and every time you see a new value, compare it to see if it's a record, and if so, remember it. Instead of initializing to the first entry like the examples, it may be convenient to initialize the max/min to placeholder values like -1 and INT_MAX (from <climits>).

Q: Can we use additional arrays for this assignment?
A: You can correctly solve this problem with only the array that is already declared in the skeleton file. However, if you want to create one more array (for a total of 2) you are welcome to do so. You should not need more than 2.

Q: Do we have to match the formatting and text exactly?
A: Generally yes. Your output must include a line containing only Nights: your-answer. The automatic grader will ignore other text. However, it is good habit to remove any debug or extra cout statements you might have added along the way before submitting your code.

Credit to David Kempe for posing a version of this problem. Assignment created by Mark Redekopp.