CSCI-15 Programming Assignment #3, due 10/14/20
Two-dimensional arrays, non-deterministic programming (100 points)

Imagine a mouse on a small island. The island is surrounded by water, except for ***EXACTLY*** 2 bridges to the mainland. There is no food on the island. Visibility on the island is obscured by fog and the mouse does not have a map of the island, so he must wander around randomly trying to get off the island via one of the bridges. If he gets onto a bridge he escapes from the island. If he drops into the water, he drowns. If he wanders for more than 100 moves, he starves to death.

Your assignment is to model this scenario. Use a map supplied in a text file to define the island. Keep an internal representation of the island's map in a 2-dimensional array. Use a second 2-dimensional array to keep track of how often the mouse moves to each location on the island. Read the island's map directly from the file using an ifstream. The file's first 2 values are the number of rows and columns in the map. The rest of the data looks like this:

`-1` – water
`0` – land
`1` – initial position of the mouse

For example, this could be a map in a data file (5 rows by 6 columns):

```
5 6
-1 -1 -1 -1 -1 -1
-1  0  0  0  0  0
-1  0  1  0  0 -1
-1  0  0  0  0 -1
-1 -1 -1  0 -1 -1
```

A bridge is land (`0`) against the edge of the map. Here, the two bridges are at `(1,5)` and `(4,3)`, counting from the upper left corner as `(0,0)`. You may assume the largest dimension of the map will never exceed 20 units (high or wide). On this map, the mouse starts at location `(2,2)`. Once you know the mouse's starting point, save it into variables and put a `0` into that position on the map. The map's values are white-space separated, so you can use >> to extract the values.

Use the random number generator `rand()` to generate the directions to move the mouse, using the values 0-3. Let 0 mean "up" or "north", toward the top of the map. Go clockwise for directions for 1, 2, and 3: "east", "south", and "west". Use a small 2-dimensional array to hold the values needed to modify the mouse's indices. The mouse movement must be done in a little function MoveMouse(), which **only** moves the mouse. I'll discuss how to do this in class.

Use the 3 maps I supply, as well as at least 2 of your own construction. Run the program 3 times on each map: once without seeding the random number generator at all, once seeding the generator with a fixed value (supplied by the user on the command line, not hard-coded) before the 100 lives, and once seeding the generator with the system time before the 100 lives. Each run comprises 100 lives of the mouse, and each life comprises no more than 100 moves (or drowning or escaping, which ends the life). You must use command line arguments to give the program the input file name, the output file name and the seeding rule. For at least one of your maps, add a lake or a bay in the island.

Output from your program must go to a text file from within the program. Before you do the 100 lives of the mouse, print the parameters supplied to your program (the input and output files and the rule for clock seeding), the map of the island and the initial position of the mouse. Then, for each life, print a noticeably different (different length) message for living (escaping), drowning, and starving (at the end of the life, not after each move!). After the 100 lives, print accumulated statistics (how many times the mouse escaped, drowned, or starved). Then, print a carefully aligned map (with constant column width, ***with the width chosen based on the width of the largest value in the count map***) showing how many times the mouse moved to each square on the island, bridge, or water. For the counts map, do not count putting the mouse down at the starting position on the map at the start of each life, only the moves. A high level design for this program might look like this:

```
parse command line parameters, open files
if needed, seed the random number generator
read the map && save initial position of the mouse and bridges
initialize the count map to all zeros
print the initialization values
loop to 100 lives
   put mouse at starting point on map
   loop until 100 moves or mouse escapes or drowns
      move mouse
      count move to target square
      if( drowned )
         handle drowning
      else if( escaped )
         handle escaping
      end if
   end loop to 100 moves, etc.
   if( starved )
      handle starving
   end if
end for 100 lives
print life statistics and count map
close files
```

Use good top-down, modular programming style for this program. Implement the various program tasks as functions, and pass in parameters as needed by the functions. **You may not use any global variables for this program – all of your variables must be local** (declared within a function). Variables needed by the whole program or by a substantial portion of the program should be declared in `main()`. Other variables should be declared locally in the functions that use them. You may (and probably will want to) use global named constants. You may also want to use structs to collect related variables (as fields) to pass around between your functions.

Email me the source, the maps you create and the output files, attached to an email. As usual, put your name and the assignment number inside the email, and put CS15, your name and the assignment in the subject line.

This program is not difficult, but will probably require three or four pages of code to complete, so START EARLY and DESIGN first, and only then write code. This program is too complex to solve sitting in front of the computer typing into the IDE editor.