Mouse Island

The purpose of this assignment is to make sure you are comfortable with two dimensional arrays, randomness, enumeration types, file I/O and following a specification.

Summary:

Your program will run the Mouse Island simulation according to the rules presented below.

You should start your program by prompting the user for an input file name.

Your program should then read the file and initialize the island. It should also check for errors in the input file.

You should then execute the simulation as many times as the input file specifies.

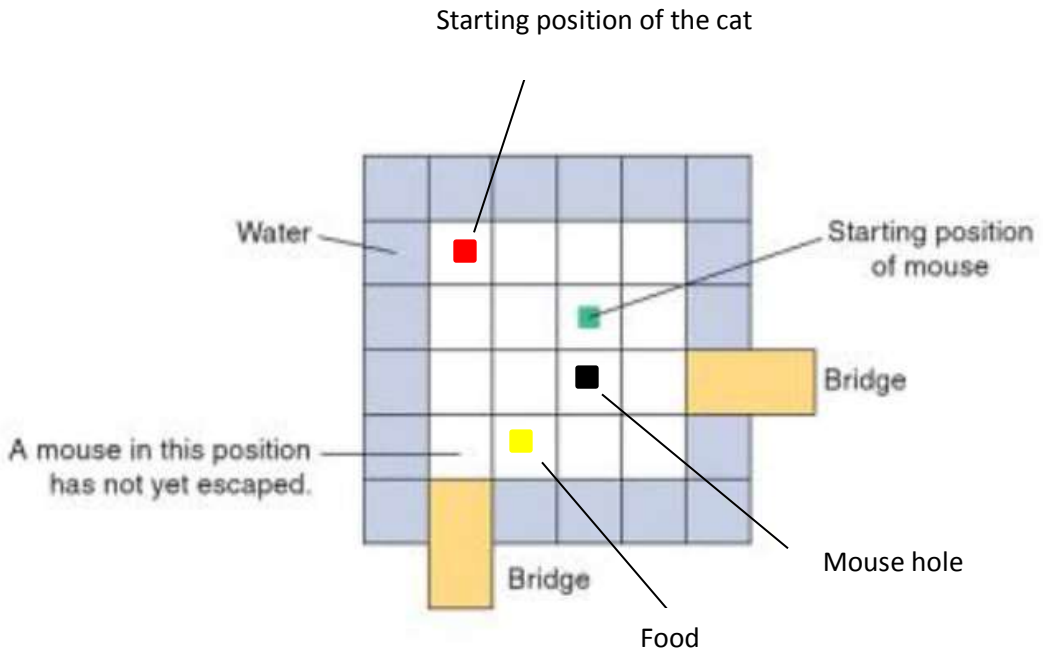
Each simulation should be presented on the screen as an animation. There should be a message explaining why the game ended with a longer delay (but not too long) printed when a simulation ends to let the user know that the simulation has started over.

Statistical information will be appended to the file MouseStats.txt before the program exits.

Guidelines:
Use enums to make the program easier to understand. Use good variable names, functions, comments and a good style. You should provide error checking on the input.

Details:

The following diagram represents a 6x6 island surrounded by water (grey-ish area).



Two bridges lead out of the island. In this example, a mouse is placed on the green square.

Write a program to make the mouse take a walk across the island. The mouse is allowed to travel one square at a time, either horizontally or vertically. A random number from 1 through 4 should be used to decide which direction the mouse is to take.

The mouse wants to escape the island because there is also a cat on the island. The cat wants to eat the mouse. The cat should move randomly in the same manner as the mouse. If the cat and mouse end up on the same square, the mouse gets eaten. The only exception to this is if the cat and the mouse are on a square containing a mouse hole. The cat cannot get the mouse while it is in the hole and continues prowling the island.

The cats do not like water, so the cat will never enter a space containing water. However, the mouse may be desperate to escape and will enter a water square. The mouse drowns when he hits the water because the current is too strong. The mouse can escape when he enters a bridge. The cat will not enter a bridge unless the mouse is in a mouse hole. If the mouse is in a hole and the cat is supposed to enter a bridge, the cat will do so and leave, seeking better prey. One the cat leaves the island, it does not return.

You may generate a random number up to 100 times. If the mouse does not find his way by the hundredth try, he will die of starvation. If the mouse finds a piece of food (yellow square), he eats it and the starvation counter should be reset, giving him more time to escape. When eaten, the food should be removed from the map. The cat has no interest in mouse food and ignores it.

The input file will be structured like so:

Map name (text)

How many full times to run the simulation (integer)

Length of the array (integer, cannot be more than 20)

Width of the array (integer, cannot be more than 20)

Next N input lines – the rows of the two dimensional array where the positions containing negative numbers represent the water, the positions in the edge containing a 0 represent the bridges, the position containing a 1 represents the starting position of the mouse, the position containing a 2 represents the starting position of the cat, positions containing a 3 represent food, positions containing a 4 represent mouse holes and all other positions contain 0s.

Note that the array will not always be square, but will not be bigger than 20 X 20.

The output file should be structured like so:

Map name (text)

How many times the simulation was ran.

The seed of the random number generator so that the user can reproduce the experiment.

A table summarizing how the game ended each time. **Example**: how many times did the mouse escape, drown, or starve. It's up to you to determine all of the ways the game can end and account for them.

A map showing the frequency of the mouse's visits to each position.