The Rules for GAME OF LIFE

The Game of Life was invented by John Conway. The game is played on a field of cells, each of which has eight neighbors (adjacent cells). A cell is either occupied (by an organism) or not. The rules for deriving a generation from the previous one are these:

Death

If an occupied cell has 0, 1, 4, 5, 6, 7, or 8 occupied neighbors, the organism dies (0, 1: of loneliness; 4 thru 8: of overcrowding).

Survival

If an occupied cell has two or three neighbors, the organism survives to the next generation.

Birth

If an unoccupied cell has three occupied neighbors, it becomes occupied.

Hint for Lab 8A Challenge:

If you need a hint for game of life, start with `Lab8Agame-Of-Life-Start.cpp`

The only thing you need to do is calculate the next generation. ADVICE: you can't do the calculation "in place". In other words, you can't modify the matrix while you are counting neighbors, because the results will be distorted. You have to use another matrix (like nextgen).

There are several ways of handling the situation. You could either make a new matrix called `neighbors[r][c]` that just records the number of neighbors of each cell for the current generation. Then you can search the `neighbors` matrix and modify your "`world`" matrix accordingly.

Another way is to make a `changes` matrix that just indicates which cells need to be turned on or turned off, then make the changes.

Still another way is to make your next generation in a new matrix called `nextgen` and for each cycle, after making `nextgen`, copy it back into `world`.

CAUTION: make sure you don't try to count neighbors on the edge cells because it will make you go outside the matrix and cause the computer to crash. So, when you count neighbors, you should use nested loops like this:

```
for (r=1; r<ROW-1;r++)
   for (c=1; c<COL-1; c++)
```

rather than

```
for (r=0; r<ROW;r++)          // causes crash when counting neighbors
   for (c=0; c<COL; c++)
```