Cellular Automata: Neighborhoods

1a. Write a program that initializes a 5x5 lattice with cell values 0-24 as shown on the transparency. Define the size, i.e. 5, in the header of your program, so that we can change it easily. Then write a function that uses as input parameters this lattice and i & j and prints for the cell (i,j) its cell value and all of its von Neumann neighbors. Use this function to print out for each cell its value and all of its neighbors. Check your program with a sketch which shows the lattice and its periodic images.

**Hint:** The modulo function % is useful to build in the periodic boundary conditions.

1b. Do the same for the Moore neighbors.

2a. Write a program that reads in an initial configuration (10x10 lattice) for the game of life. Use the file

`kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/init_10x10_rand.data`

Write a function which finds for a given cell (i,j) its number of von Neumann neighbors. Test your function with the same initial configuration, read in a cell (i,j) and print out the number of neighbors.

**Hint:** To write the function which finds the number of neighbors, use a variation of your function in 1a.

2b. Do the same for the Moore neighborhood.
Game of Life

Today you will write your own game of life program!

3a. Write a program that reads in the initial configuration (5x5 lattice) from file:
   "kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/init_5x5_rand.data"

**Hint:** Use your program of exercise 2b (Do not remove the functions which calculate the number of neighbors, because you will need them in 3b.)

3b Add the main part of the game of life program: the loop over the cells and the update of the cells (see flow chart). Print out the new configuration. Check your program with the result with the transparency (or
   "kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/ame_of_life_5x5"
)

3c Add the time loop (see flow chart) and check your result after 10 timesteps with the transparency (or file given in 3b).

4. Do the initialization in a function. This will be useful for Wednesday, because we will then play with different initial configurations. We will watch the resulting patterns with graphics!

**Solutions to Exercises 1. and 2. (class of Feb.2):**
   "kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/game_ex1.cc"
   "kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/game_ex2.cc"
**Game of Life**

5. Let us see how you can watch a movie of the game of life. Use your program from exercise 4. and change the output such that it looks like the print out on transparency. So add at the beginning of each configuration two lines, the first with “#pause 4” and the second with “#string time = timevalue”. Add after each configuration a blank line. Then look at the resulting output. In case you print on the screen then use:
   ```
   executable | DynamicLattice -nx 5 -ny 5 -matrix
   ```
or in case your output is in a file:
   ```
   DynamicLattice -nx 5 -ny 5 -matrix < fileout
   ```
   Change the number “4” to learn about its effect. The “5” is specifying the size of the lattice in the horizontal and vertical direction.

6. Now start with different initial configurations. Use a 30x30 lattice with all cells dead but the pattern of Fig.1a of alive cells. Watch the pattern how it changes with time. Repeat this for the patterns 1b - 1g. (Use the Moore neighborhood.)

7. Next you will learn how to start with a random configuration, i.e. each cell is randomly chosen to be alive or dead. To do so use the program
   ```
   ~ kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/bit_rand.cc
   ```
   and run it so see what it does. Now use this program to write your own random initial configuration. Run it.

**Solutions to Exercises 3. and 4. (class of Feb.2):**

   ```
   ~ kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/game_fx3.cc
   ```
   ```
   ~ kvollmay/classes.dir/capstone_s2001.dir/cellular_automata.dir/game_fx4.cc
   ```