Game Of Life

1. Initialization And Printing

Write a program that initializes a 5x5 lattice as follows:

0	0	0	0	0
0	1	1	1	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Define the size, i.e. 5, as constant in the header of your program, i.e. before main. This will allow us later to change the size of our lattice easily. Then print the lattice on the screen. That means that you will get on the screen the lattice as shown above.

2. Von Neumann Neighbors

2a. Determine Neighbors

Next you determine the von Neumann neighbor lattice sites and their values. To be able to test our program initialize the 5x5 lattice as follows:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Print out the lattice (same as in 1. but for this lattice). Now read in two integers `isite` and `jsite` which specify the site for which you try to determine its von Neumann neighbors. Print out the lattice values for the site `(isite,jsite)` and the lattice values of the right,left, top and bottom sites. Check your result for different values of `(isite,jsite)`. Be careful to use periodic boundary counditions. (Hint: To take into account the periodic boundary conditions use either `if, else` etc. or to shorten your program use %.)

2b. Number of Neighbors

You are now ready to determine the number of neighbors (=number of living neighbors). Change your program such that it reads in the initial configuration (5x5 lattice) from the file ¨ kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/init_5x5_rand.data
The program then prints the lattice and reads in a lattice site `(isite,jsite)`. For this site the program determines the number of living von Neumann neighbors and prints out the result.

2c. Do the same as in 2b. but write a function which has as input the lattice and returns the number of living neighbors.
Hint: To declare (and similarly to define) the function use for example:

```
int vonNeumann (const int lattice [][LATSIZE],int isite,int jsite);
```

Game Of Life

2. Von Neumann Neighbors

2b. Number of Neighbors

Using your result or the solution to 2a. change the program such that it reads in the initial configuration (5x5 lattice) from the file

¨ kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/init_5x5_rand.data

The program then prints the lattice and reads in a lattice site `(isite,jsite)`. For this site the program determines the number of living von Neumann neighbors and prints out the result.

2c. Do the same as in 2b. but write a function which has as input the lattice and returns the number of living neighbors.

Hint: To declare (and similarly to define) the function use for example:

```
int vonNeumann (const int lattice [][LATSIZE],int isite,int jsite);
```

3. Finish Game of Life Program

3a. Use your program from 2c. Add the main part of the game of life program: the loop over the cells to determine each new cell value. After this loop update all cells at once. (See Flow Chart) Remember that you need for this two two-dimensional arrays. For example if your lattice is called "lattice" then while you determine the new lattice sites determine the neighbors with lattice, but write the new cell values into another array, e.g. "newlattice". Check your program with the result for t=1 in the file

kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game3b_data

3b Add the time loop (see flow chart) and check your result after 10 timesteps with the result in game3b_data.

4. Graphics

Let us see how you can watch a movie of the game of life. Use your program from exercise 3b. and change the output such that it looks on the screen like the file

kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game4_data

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.

Solutions:

kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game2a_ifelse.cc

kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game2a_mod.cc

Game Of Life

3. Finish Game of Life Program

3b Use your program for 3a. or the solution listed below. Add the time loop (see flow chart) and check your result after 10 timesteps with the result in
kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game3b_data

4. Graphics

4a Let us see how you can watch a movie of the game of life. Use your program from exercise 3b. and change the output such that it looks on the screen like the file
kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game4_data
So add at the beginning of each configuration two lines, the first with "`#pause 2`" 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 "2" to learn about its effect. The "5" is specifying the size of the lattice in the horizontal and vertical direction.

4b Now let's watch a movie of a 10x10 lattice. Read in the initial configuration
¨ kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/init_10x10_rand.data
and change in your program the lattice size from 5 to 10. Rerun your program and adjust the command of Dynamic Lattice accordingly.

Solution:
˜ kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game3a.cc

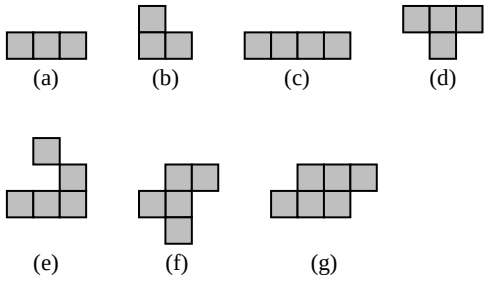

Fig.1

February 11, 2003 CAPS 493-24 Spring 2003

Game Of Life (Advanced)

5. Moore and Patterns

5a. Use your program of 4., that means start with the 5x5 lattice of file
¨ kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/init_5x5_rand.data
and do 10 timesteps. Instead of the von Neumann neighbors use Moore neighbors (add another function). Check your result with the file
k̃vollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/game5a_data

5b. Now start with different initial configurations. Use a 30x30 lattice with all cells dead but approximately in the middle of the lattice the pattern of Fig.1a of alive cells. Using Dynamic Lattice watch the pattern how it changes with time. Repeat this for the patterns 1b - 1g. (Use the Moore neighborhood.)

6. Parity Rules

Before you work on this program, please let me know, so that I can explain some background. Next we change the rules of the game of life. Use the following "Parity Rule": For a given site count how many living von Neumann neighbors the site has, add to the number of living sites the number of the lattice value at the site itself. If the resulting number is even, then the new site value is 0. If the resulting number is odd, then the new site value is 1.
Use a large lattice (e.g. 100x100) and start with all cells dead but a square (2x2 or larger) of living cells in the middle. Run your program and look at it with DynamicLattice.

Random Numbers

1. Uniform Random Numbers

1a. Copy the program
~ kvollmay/classes.dir/capstone_s2003.dir/game_of_life.dir/float_rand0-1.cc
into your working directory. Compile the program and let it run. Have a look at the program to learn how to get random numbers. Whenever you want to write a program in which you use random numbers you need to do the following step:
1. Include in the header of your program the two lines:

```
double randomd(long *);
   long idummy = -7;
```

2. Include at the end of your program the definition of the function randomd, i.e. lines 21–49 of float_rand0-1.cc.
3. whenever you want another random number use
`randomd(&idummy)`

1b. Have a look at which random numbers you get: To do so set up the following unix-command:
`executable | awk '{print NR,$0}' | xgraph -m -nl`

2. Probability

2a. Write a program that prints 500 numbers. Each number is with probability $p = 0.3$ "1" and otherwise "0". Run the program and check it with xgraph as in 1.

2b. Change your program from 2a. such that it reads in from screen the probability p and the number of random numbers N. The program then produces N numbers, where each number is with probability p "1" and otherwise "0". The program counts how many of the drawn numbers are "1" (n_{alive}) and prints out $(n_{\text{alive}})/N$. Run your program a few times for different values of p and N.

3. Integer Random Numbers

Use the random numbers from 1. to print out 500 random numbers which are with equal likelihood $v = 0, 1, 2, 3, \ldots, 9$ or 10. (We will use this method for the traffic model to draw velocities for cars on the road.) Check your program with xgraph as you did in 1b.
Once your program is working generalize 10 to the constant `VMAX=10` which you define at the beginning of your program.

Random Numbers And Game Of Life (Advanced)

1. Work on 1a,b. and 2a,b on the backside of this page.

2c Write a program which allows you to make a graph of $(n_{\text{alive}})/N$ as a function of N. Keep $p = 0.3$ constant. So your program should print out two columns, where the first is N and the second is $(n_{\text{alive}})/N$. (Vary N between 0 and 50000.) Look at your result with
`executable | xgraph -m`
Look at the result for different values of p.

3. Work on 3. on the backside of this page.

4. Game of Life: Density As Function of Time

4a. Use your program for the game of life (5a.) Write a function which initializes your lattice as follows: Each cell is alive with a probability p_{alive}. Initialize your lattice by calling this function for a specific p_{alive}, e.g $p_{\text{alive}} = 0.5$.Test your program with a 5x5 lattice by printing out only the initial configuration.

4b. Use your program for 4a. with $p_{\text{alive}} = 0.4$ and now for 2000 timesteps and a 100x100 lattice. Change your program such that for each time step t the program prints out t and (number of alive cells)/(total number of cells)$= n_{\text{alive}}/N$. Plot the resulting n_{alive}/N as a function of the time t.

4c. Print out t and n_{alive}/N only every 20 timesteps and run your program for 3000 timesteps. Vary i) the lattice size ii) p_{alive} and iii) the neighborhood. Draw the different curves for i) all in one graph. To do so redirect your output into files, for example

```
executable > lat100p04
```

and look at the different files with for example

```
xgraph -m lat*p04
```

Do the same for ii) and iii).

5. Equilibrated n_{alive}/N As Function of p_{alive}

Use a 100x100 lattice, 1000 timesteps and the von Neumann neighborhood. You saw in 4. that the density n_{alive}/N after a certain number of timesteps no longer changes. Add to your program that you loop over p_{alive} and that your program prints out p_{alive} and the density only for the lattice after the last timestep. You obtain the density as a function of p_{alive}. Look at your result with xgraph.