Name:\_\_Xiang, Xin\_\_ Date: \_\_\_\_ \_Apr 18, 2020\_ \_ \_

Section: \_ 005\_ \_ \_ Github Username: \_\_FreyaXiang\_ \_

**Lab 4**

Total in points (100 points total):

Professor’s Comments:

Honor Pledge: I have neither given nor received aid on this assignment.

Signature: Xin Xiang

The general logic of this program is to create nested loops.

The number of generations is passed to the main function as argv[1]. This is the number of times the outer loop iterates.

The name of the file with the initial matrix is passed to the main function as argv[2]. Open this file and read the data into an array by linear fashion. We call this array “initial”.

Also, create another array with the same size and call it “result”.

Therefore, the neighbors of a cell with index m in “initial” are:

|  |  |  |
| --- | --- | --- |
| m-6 | m-5 | m-4 |
| m-1 | m | m+1 |
| m+4 | m+5 | m+6 |

Then the pseudocode of the nested loops is:

for each generation {

for each cell with index m {

int count=0;

Check how many living neighbors are around each cell: If the index of a neighbor

is in the range [0, initial.size-1] and that neighbor is alive, count++.

Then check the state of the cell and update the state into “result”:

if the cell alive (initial[m]==1) {

if count<2 or count>3 {the cell becomes dead, result[m]=0}

else {the cell remains alive, result[m]=1}

}

if the cell dead (initial[m]==0) {

if count==3 {the cell becomes alive, result[m]=1}

else {the cell is still dead, result[m]=0}

}

}

Copy data from “result” to “initial” and a new iteration begins.

}

In the end, write “result” matrix into the output file and close all the files.