Programming Project 1

Conway's Game of Life

Overview:

The goal of the project is to organize the code according to the multiple file etiquette we discussed in class. Which involves separating the single provided source into the specified files.

There might be many language details that are unfamiliar to you. That is completely fine, we will discuss them in time, and **you do not need to know anything about them to complete this project**. In addition to File Etiquette and Class Implementation, there are several aspects of programming that this project stresses.

- Working with unfamiliar code and being able to get a general understanding of the overall flow of execution. Using the IDE's debugger can help in this regard using breakpoints.
- Figuring out what is being specified, and for what aspect of the project the given information relates. Read everything first and devise a plan, as you become more familiar with what you need to achieve modify the plan accordingly.

Organize the code:

Following what we discussed on multiple file etiquette take the single source file provided, and divide it into appropriate header files and implementation files, one pair of files for each class. Place the main routine in its own file named `main.cpp`. Make sure each file #includes the headers it needs. Each header file must have include guards. Only include header files when definitions are actually needed; forward declare objects in situations when the compiler only requires to "know" about the objects existence but not it's definition.

Now what about the global constants? Place them in their own header file named `globals.h`. And what about utility functions like `delay`, `clearScreen` , or report? Place them in their own implementation file named utils.cpp, and place their prototype declarations in `globals.h`.

The first thing you should do is make sure that the single source compiles as is. Play around with it to get comfortable the program. The program implements Conway's Game of Life https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life, which is a simple simulation where cells are either alive or dead based simple rules based on under population or overcrowding.

Analysis:

Get a high level understanding of the behavior of the program (You do not need to know anything about inheritance or polymorphism at this point). How do we go about determining if a "birth" has occurred in any particular cell of the world? In other words, using the information we have available (variables) what is the logic in identifying a birth occurring in any cell on any day? Note, that a cell being alive on any particular day does not imply that was birthed. In what class and in which member function of that class can we obtain that information?

You should notice that the utility function report does not do anything useful other than printing "Hello World!" to the console. Replace "Hello World!" with a brief statement (a few sentences) with your answer to the above questions.

Submission:

Submit via canvas one zip file containing **only** the 13 source files produced for this project do not include any IDE specific project files:

life.h	blinker.h	glider.h	world.h	game.h	globals.h	
life.cpp	blinker.cpp	glider.cpp	world.cpp	game.cpp	utils.cpp	main.cpp

The zip file should be named <lastname>_<id>.zip; for example nguyen_123456.zip.

If I take these 13 files, I must be able to compile them using VS2017 without any errors or warnings. Be sure to you do not introduce any compilation or linking errors. **Work incrementally**, compile often and submit regularly, your previous submissions will be overwritten on canvas. What you submit must compile, incomplete code that works is always better than any code that does not compile regardless of how "correct" most of the implementations are.

Additional Considerations

- Other than in the function report, you should not make any actual changes to any of the functions or classes, you just need to move them around. The word friend and the word sequence pragma once must not appear in any of the files you submit.
- Your program must not use any global variables whose values may change during execution. Global constants (e.g. MAXROWS) are all right.

Hints:

- Start with making sure the original source compiles, then spend some time playing around with the code to try to get a sense of what is going on. Feel free to go cray, you can always re-download the source. Start at the "top", the main function. See what objects are created and then take a look at those objects' construction/member function calls. It is ok if you don't completely understand what is going on, there is some funky inheritance action going which we'll discuss in detail later.

- If we replace your main.cpp file with the following, the program must build successfully under both Visual C++ and clang++:

```
#include "game.h"
#include "game.h"
#include "world.h"
#include "world.h"
#include "life.h"
#include "life.h"
#include "blinker.h"
#include "blinker.h"
#include "glider.h"
#include "glider.h"
#include "globals.h"
#include "globals.h"
int main(){}
```