Programming Project 1

Conway's Game of Life

*Read everything carefully, most questions I get have answers within this specification.

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 its definition.

Now what about the global constants? Place them in their own header file named `utils.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 `utils.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. For those who are feeling adventurous look for some Life patterns and try to implement them.

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). Consider the Worlds member variables: m_grid_1, m_grid_2, and m_toggle.

How might you change the program so that World only has one grid as member variable and none others, and have exactly the same output? You may have as many local variable as you need, but no

global or additional member variables in any object. You do not need to actually implement this, although you can for you own enrichment; just provide a brief description.

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 discussion addressing the queries above. This may require stepping through the code to get an overview of execution, hand tracing is an option but using the debugger with break points is much more efficient.

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	toad.h	ring.h	world.h	game.h	utils.h	
life.cpp	toad.cpp	ring.cpp	world.cpp	game.cpp	utils.cpp	main.cpp

I am expecting .h and .cpp source file extensions, make sure you're file names conform to the given specification. Mac OSX adds some files when you generate zip archives, those are fine.

If I take these 13 files, I must be able to compile them using VS2017/2019 without any errors or warnings.

Additional Considerations

- **Work incrementally/iteratively**, this is life advice for all programming in all situations. Making large sweeping changes to any code base without validating is the extremely counterproductive. For this project specifically, start with just moving only one class definition into its own header file, reconcile what needs to be reconciled and compile. If successful, only then go onto moving more code, e.g. that class' implementation, rinse/repeat.
- 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.
- 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 ok.
- 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

```
#include "game.h"
#include "game.h"
#include "world.h"
#include "world.h"
#include "life.h"
#include "life.h"
#include "toad.h"
#include "toad.h"
#include "ring.h"
#include "ring.h"
#include "utils.h"
#include "utils.h"
int main(){}
```

- If we replace your main.cpp file with the following, the program must **not** build successfully

```
#include"game.h"

int main() {
  World w;
  Game g(nullptr, 0);
}
```

- If we replace your main.cpp file with the following, the program must **not** build successfully

```
#include"game.h"

int main() {
      Life l;
}
```

- There are several other tests similar to those above to ensure proper file etiquette, you are encouraged to think of more and share them.