Nicole Parker
Assignment 1

Design:

For this program I decided that although using a 2D array may be more efficient a vector would better represent an infinite space. I will focus on cells and their surrounding area to determine the programs actions. I will use a struct for cells that will hold cell coordinates.

1. vector <cell> livecells
   vector <cell> emptycells
   struct cell:
      int x
      int y
      Bool killMe
      Bool fillMe

2. Iterate through livecells
   A. Check surroundings spaces for livecells

      -if empty, check emptycells and if that coordinate pair does not exists in emptycells add it.
      - if not empty increase neighbor count
   B. If neighbor count > 3 killMe = True
C. If neighbor count < 2 killMe = True

   A. Check surrounding space for livecells
      -If neighbor count == 3 set fillMe == True

4. Iterate through livecells
   A. If killMe == True then remove from livecells

5. Iterate through emptycells
   A. If fillMe == True then add to livecells
6. Clear emptycells

Testing:

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Selection of pattern in range	Input between 1-3	main() gameOfLife.setup()	Ask user for x coordinate	Ask user for x coordinate
Selection of	Input between	main()	Tell user input	Tell user input invalid.

pattern out of range	< 1	gameOfLife.setup()	invalid. Ask user to input x coordinate	Ask user to input x coordinate
Selection of pattern out of range	Input between >1	main() gameOfLife.setup()	Tell user input invalid. Ask user to input x coordinate	Tell user input invalid. Ask user to input x coordinate
Selection of pattern with a non-digit	Input that is not a digit	main() gameOfLife.setup()	Tell user input invalid. Ask user to input x coordinate	Tell user input invalid. Ask user to input x coordinate
Selection of x coordinate with a non-digit	Input that is not a digit	main() gameOfLife.setup()	Tell user input invalid. Ask user to input x coordinate	Tell user input invalid. Ask user to input x coordinate
Selection of y coordinate with a non-digit	Input that is not a digit	main() gameOfLife.setup()	Tell user input invalid. Ask user to input y coordinate	Tell user input invalid. Ask user to input y coordinate
Selection of oscillator pattern	Input of 1	Main() gameOfLife.start()	Output press enter to start. Draw oscillator	Output press enter to start. Draw oscillator
Selection of glider pattern	Input of 2	Main() gameOfLife.start()	Output press enter to start. Draw glider	Output press enter to start. Draw glider
Selection of glider gun	Input of 3	Main() gameOfLife.start()	Output press enter to start. Draw glider gun	Output press enter to start. Draw glider gun
Selection of continue	Input of c or C	Main() gameOfLife.start()	Output continue for 50 generations	Output continue for 50 generations
Selection of quit	Input of q or Q	Main() gameOfLife.start()	Output “Good Bye” end program	Output “Good Bye” end program
Selection of continue with invalid input	Input of a digit	Main() gameOfLife.start()	Invalid input prompt to re-enter input	Invalid input prompt to re-enter input
Selection of continue with invalid input	Input of a letter other than c or q	Main() gameOfLife.start()	Invalid input prompt to re-enter input	Invalid input prompt to re-enter input

<u>Reflection:</u>

It took a while to settle on a design for this program. I was having a hard time determining how to handle cells and then how to handle their neighboring cells. I found that it was not necessary to keep track of every cell on the board. The best designing I found was to keep track of only the live cells, essentially creating multiple mini boards for each cell. These mini boards consisted of a cell and its neighbors and they could then be addressed appropriately by whatever function that was needed. Ultimately I decided to use vectors instead of 2D array so I would be able to better represent infinite space. This would make the issues of the edges and outermost corners a non-issue. I did run into a

problem with vectors. I was not aware that when using the erase function with vectors the index would then become inaccurate. The erase function invalidates any iterators at or after that point. I was able the find a resource that suggested to use a remove it function instead (I included a link to the reference I used.). This function was much more efficient. The errors related to the erase function were very confusing. The program worked for the oscillator for two generations and then stopped working, which at first just made no sense. I had to look at each generation and every cell that was supposed to be killed, or filled and compare it to what actually happened. Eventually this lead me to think about the possibility of the index being off, but this took a really long time to get to. I had a few other minor issues of setting a boolean to true when it should have been false which created an infinite loop but this was quickly fixed.

http://stackoverflow.com/questions/9053883/eraseremove-if-anomaly