**Algorithm approach:**

**Not optimal:** Brute Force approach could be registering each cell status (dead or alive) and at each iteration go through each cell and check its neighbors and update accordingly. Given the size of problem 64 bit coordinate system it is not feasible as it will require huge memory to store cell information and huge runtime to check each cell.

**More optimal approach**: Given the number of seed input are way less and at any given time alive cells are way less than dead cell in the given world size I am using approach to register only alive cells. This gives huge benefit of problem becoming irrelevant of alive cell location as the problem is now local in approach.

During each iteration we make 2 passes.

We query the alive cell for its neighbors and maintain a gestation list (likely to be alive if their alive neighbors are more than as needed by rule). Also we update no. of living cells to indicate their living neighbors count.

During the second pass for each alive cell and each gestation cell we have count of no. of living neighbors. Based on rulebook and number we either kill living cells or bring gestation cells to life.

We clear off gestation list and start over again and only maintain current living cell list (similar to initial seed list)

**Functions and Control**

The program provides basic simulation using OpenGL that represents alive cells at each iteration.

Since the world size is large, to navigate and see life in other parts there are some basic keyboard controls added.

Use Arrow keys to pan left/right/top/bottom

Use +/- to zoom(2x) in and Out

**Running the program**

GameOfLife.exe can be run through command prompt with argument of test file path. If no path is provided it by default looks at tests/Input.txt

***Sample command: C:\Users\vgupt11\Desktop\Conway>***GameOfLife.exe tests/Input4.txt

Makefile command inside mingw shell: mingw32-make