**Purpose/Overview**

* A program that simulates Conway’s game of life.
* The cells are on a 32x12 grid.
* Live cells are represented by the ‘@’ character while dead cells are represented by blanks.
* Live cells with fewer than two living neighbors die of under-population.
* Live cells with more than three living neighbors die of over-population.
* Other live cells continue to the next generation.
* Dead cells with exactly three neighbors become alive again.

**Requirements**

* The evaluation of sell states must be based completely on the current state
* Must provide a menu choice to:
  + Allow the user to select which action they would like to perform.
  + Clear the grid.
  + Exit the simulation.
  + Start/resume the simulation for a user-specified number of generations.
  + Single-step the simulation.
  + Toggle states of one or multiple cells.
* Must be in a stopped state initially.
* Must run at one simulation a second.

**Classes**

Grid:

* Stores the current state of the grid.

Data members:

* bool grid[12][32]: private, stores current state as a 2-d array of Booleans
* int gen: private, stores generation number
* bool compact: public, stores the display mode

Methods:

public:

* void next(int steps): Progress simulation
* void display(): To display the grid
* void toggle(int, int): To switch the life status of a cell
* void reset(): Reset grid
* void save(ofstream&): To file
* void load(ifstream&): From file

private:

1. void next(): Progress simulation by a step
2. int countLiveNeighbours(int, int): Number of live neighbours of each cell

**Global Data/Functions**

1. No global variables
2. Functions in main.cpp:
   * void toggle(Grid&): Switches the state of cells
   * void save(Grid&): Saves state to file
   * void load(Grid&): Reads state from file
   * void tokenize(const string&, vector<string>&, const string&): isolates integers and adds to vector

**High-level Architecture:**

Algorithm:

1. Create a Grid object
2. Ask the user to pick an option
3. If the user chooses 8, quit.
4. Otherwise, use the corresponding function call provided by the grid object to execute the user’s choice.
5. Repeat from step 2.

Note: Even though the spec uses column-major notation, the program follows the convention of using row-major code internally. However, this is invisible to the user who uses column-major notation to interface with the program.

**User Interface**

The program uses a text-based, menu driven interface.

The grid is represented pseudo-graphically, using ‘@’ to represent live cells and ‘ ‘ to represent dead ones.

Sample:

Generation: 1

1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

0 | |@| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

1 | | |@|@| | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

2 | |@|@| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

The user enters option numbers as ints.

Test cases:

* Input validation: Handle abnormal choice inputs like 23.
* Note: only basic validation implemented as required by spec sheet.
* Toggle cells with random inputs within range.
* Works with provided sample of (0,1) (1,2) (2,0) (2,1) (2,2), glider works as specified
* Also tested with various still lives and oscillator patterns