**Purpose/Overview**

To run a simulation of Conway’s game of Life.

**Requirements**

* The program must allow the options of going ahead by one or more steps, going back by one or more steps.
* The grid must be toroidal.
* Must allow command-line arguments.

**Classes**

Grid:

* Stores the state of all cells on the grid.
* Data members:
  1. int sleepTime: stores the pause time between displays.
  2. vector< vector<bool> > grid: stores the state of the cells
  3. int gen: stores the current generation number
  4. int R, C: stores the number of rows and columns
  5. vector<string> stack: stores the past states of the grid
  6. bool compact: stores the print mode
* Methods:
  1. Grid(int r, int c, int t): parameterized constructed
  2. void next(): progresses by one step
  3. void next(int steps): calls the next() method steps times, and pauses sleepTime milliseconds.
  4. void display(): prints the grid and generation number
  5. void toggle(): flips the state of a cell
  6. void reset(): Set grid to initial state

**Global data/Functions**

* No global data
* Important functions in the main file:
  1. int main(): displays menus and handles choices
  2. void toggle(): parses user input and flips cells

**High-level architecture**

* The main.cpp file is responsible for the creation and management of Grid objects
* Grid objects are defined in the grid.cpp file
* The Grid object provides an interface for all the important operations outlined by the requirements
* The high-level algorithm is as follows:
  1. Create a grid object and initialize the values to those accepted as command line arguments.
  2. Display the options.
  3. Accept the user input.
  4. Quit if the user chooses to quit.
  5. Otherwise, call the corresponding function of the Grid class.
  6. Go back to 2.

**User Interface**

* The program uses a command-line interface to interact with the user.
* The program supports a pseudo-gui ‘character graphics’ display of grid states
* The user optionally enters the width, height and sleep interval as command-line arguments.
* The user also enters the option numbers.
* Some options require additional input:
  1. The multiple forward and multiple back options require input of number of steps.
  2. The toggle states option requires input of the set of cells to flip.

**Test cases**

* The system is first tested with abnormal command line arguments:
  1. Invalid inputs to each parameter(such as ‘–h=-1’) are successfully caught by the system.
  2. Badly formatted arguments( such as ‘=hjs-7’) are successfully caught by the system
* The individual options are then tested:
  1. Toggle cells (0,1) (1,2) (2,0) (2,1) (2,2). This is a glider configuration.
  2. Go forward one step.
  3. Go back three steps(Program catches erroneous input and issues warning)
  4. Go forward 7 steps.
  5. Go back 3 steps.
  6. Go forward 9 steps.
  7. Reset.