200 points. Individual Work Only.

Objectives:
To design and implement an efficient multithreaded version of the "Game of Life" program using OpenMP.

Problem Statement:
The objectives of this homework are:

1. Design and implement a multithreaded version of the Game of Life program (developed in Homework-1) using OpenMP. The program must take the problem size, maximum number of iterations, and number of threads as command-line arguments. Make sure the program checks if there is a change in the board state between iterations. The program must create a parallel region only once, *i.e.,* you should not create a new thread during each iteration.
2. Test the program for functionality and correctness (the output from the multithreaded version must be identical to the single threaded version). Describe how you tested the multithreaded version for correctness in your report.
3. Measure the performance of the program and optimize the program to improve performance. Please make sure to use the Intel Compilers for best performance.
4. Determine speed and efficiency of the multithreaded version of the program, plot the performance results, analyze the performance results, compare the performance with the sequential version, and include the analysis in the report.
5. **Graduate Students only (both CS 632 and CS 732)**: Implement a two-dimensional distribution of the data using OpenMP and measure the performance for the following grid sizes: 1x16, 2X8, 4X4, 8X2, and 16X1. Analyze the performance across different grid sizes and include this analysis in the report.
6. **Ph.D. Students only**: Test your program using the gcc compiler with appropriate optimizations flags for the various problem size, maximum number of iterations, and number of threads as described below. Compare and analyze the performance of your program with the performance results obtained using the Intel compilers and include this analysis in your report.

Guidelines and Hints:

1. Review *Chapter 5. Shared Memory Programming with OpenMP* in the textbook, download the source code from the textbook website, compile and test the programs on a Linux system (you can use elvis machines in CS for testing). You can also review the OpenMP Tutorial at https://computing.llnl.gov/tutorials/openMP/ and the Introduction to OpenMP Tutorial at https://www.openmp.org/uncategorized/tutorial-introduction-to-openmp/.
2. Test the multithreaded version of the program for small problem sizes (say, 5x5 or 10x10) first by comparing the result with the sequential version (you have to use the same seed in both versions of the program) on the CS Linux systems. After you are convinced that your program is working correctly, execute your program on the ASC cluster for 1, 2, 4, 8, 10, 16, and 20 threads and note down the time taken. Use the matrix size 5000x5000 and maximum iterations as 5000 for all the test cases.

Compute the speedup and efficiency and plot them separately. Include details on how you computed the speedup and efficiency in the report. Compare the performance on the OpenMP version with the sequential version and include your analysis in the report.

3. Make sure that you use elvis machines in the CS department for all development and testing and **for obtaining the performance results**. When you are testing, you can initialize the matrix and print the output only in the main thread.
4. Try to use *run_script* to run your tests.
5. Make sure you comment out any print statements you might have to print the board when you execute with larger problem sizes. Also, execute the program three times and use the average time taken.
6. Check-in the final version of your program and the output files to the *git* server and make sure to share your *git* repository with the Instructor. Please make sure you are not writing the array to stdout, this would result in large job output files.

Program Documentation and Testing:

1. Use appropriate class name and variables names.
2. Include meaningful comments to indicate various operations performed by the program.
3. Programs must include the following header information within comments:

```
/*
  Name:
  RowanID:
  Homework #:
  To Compile: <instructions for compiling the program>
  To Run: <instructions for running the program>
*/
```

Report:

Follow the guidelines provided in Canvas to write the report. Submit the report as a Word or PDF file. Please include the URL to your *git* location in the report and make sure that you have shared your *git* repository with Instructor. If you are using specific compiler flags, please make sure to include that in your report as well as README.md file and check-in the README.md file to the *git* repository.

Submission:

Upload the source files and report (.doc or .pdf file) to Canvas in the assignment submission section for this homework. Do not upload zip/tar files, upload individual files.