**CSCI 235 – Software Design & Analysis II**
**Assignment 2**

## *Introduction*

Before starting this assignment, read the following programming rules:
http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci235/programming_rules.pdf

This assignment tests your ability to design and implement recursive and backtracking algorithms. The program must compile and run on the server eniac.geo.hunter.cuny.edu in G-Lab. You must work on this assignment individually.

Your grade will be based on the following:
50% = Correctness (conformance to the requirements below)
    The program must compile and run on one of the G-lab machines. In addition, it must perform the functions outlined in the "Assignment" section.
25% = Design
    The program must show a reasonable object-oriented decomposition of the assignment into classes and must be recursive.
15% = Performance
    The implementation must be as efficient as possible in terms of the amount of memory used and in terms of the amount of computational cycles used.
10% = Documentation and style
    The implementation must have good comments; variables must have reasonable names, and the submission must have instructions on how to compile and execute (run) the program. Prompts, if any, must be clear and must explain what the program expects and requires to execute successfully. The package must be placed on Blackboard as outlined in the "Submission" section.
There is a 10-point late penalty per day after the first day.

## *Background*

Sudoku is a logic-based, combinatorial number-placement puzzle. For example:

```
 =============================
| 4  9  1 | 2  3  7 | 5  6  8 |
| 5  7  8 | 1  9  6 | 2  3  4 |
| 6  3  2 | 4  5  8 | 7  9  1 |
 =============================
| 8  1  6 | 7  4  3 | 9  5  2 |
| 2  4  7 | 5  8  9 | 6  1  3 |
| 9  5  3 | 6  1  2 | 4  8  7 |
 =============================
| 1  2  9 | 3  6  4 | 8  7  5 |
| 3  8  4 | 9  7  5 | 1  2  6 |
| 7  6  5 | 8  2  1 | 3  4  9 |
 =============================
```

The objective is to fill a 9x9 grid with digits so that each column, each row and each of the nine 3x3 sub-grids that compose the grid contains all the digits from 1 to 9. Duplicate numbers may not appear twice in the same row, column or in any of the nine 3x3 sub-grids of the 9x9 playing board. Above is one example of a valid Sudoku board.

Wikipedia has a page ( http://en.wikipedia.org/wiki/Sudoku ) about Sudoku puzzles.

## *Assignment*

Write a program to generate valid Sudoku boards. Your program should begin by placing a random number between 1 and 9 in the top left cell of the grid. It should then place other numbers in the remaining cells to satisfy the constraints of the puzzle. Your solution must be recursive. It may also use a backtracking algorithm.

## *Submission*

Submit your source code on Blackboard. Submit all your code and documentation as one “tar.gz” file. A tar file concatenates a bunch of different files (without compressing them). A gz (gzip) file compresses a single file. You can create a tar file (named "a2.tar") in the same directory from three files (main.cpp This.hpp This.cpp) with the following command:

```
tar -cvf ./a2.tar main.cpp This.hpp This.cpp
```

You can gzip the tar file above with the following command:

```
gzip a2.tar
```

This creates a file in the same directory called "a2.tar.gz", which is what you should submit on Blackboard.