Homework Assignment #3 (100 points, weight 15%)
Due: November 23 at 10:00 a.m. (in lecture)

Guidelines for programming parts: Write your program in some high level programming language such as C, C++, Java. Hand in pseudocode, program and output results (note if too many tests are done, print only a sample of output results and summarize results in tables). Please, specify the platform you run your tests on (machine speed, machine RAM and operating system).

SUDOKU by heuristic search
Refer to assignment#2 for introduction to the game **SUDOKU**. In this assignment you will develop a simulated annealing and a tabu search method for completing the SUDOKU puzzle.

1. (50 points) **SUDOKU by simulated annealing** In this problem you will implement and test a simulated annealing algorithm to solve SUDOKU based on the paper:

   Rhyd Lewis, Metaheuristics can solve sudoku puzzles, *Journal of Heuristics* **13**, 2007.

   The input for your algorithm is a SUDOKU puzzle (a sudoku grid partially filled by the "givens").

   Abide by the following requirements:

   - Initial solution: randomly fill out each of the 3×3 squares in such a way that it contains each number in $\{1, 2, \ldots, 9\}$ exactly once. Moves will never change the frequency of numbers within each 3×3 square.
   - Each solution considered throughout the search has each 3×3 subsquare filled containing each number of $\{1, 2, \ldots, 9\}$ exactly once. Rows and columns may not satisfy the desired property of having each number appearing exactly once.
   - Here we specify the cost function to be minimized (replace maximization of profit by minimization of cost function). For each row i (column j) the row score rs_i (column score cs_j, respectively) is the number of symbols absent in row i (column j, respectively). The total cost of a solution is the sum of the scores of its rows and columns, that is $(\sum_{i=1}^{9} rs_i) + (\sum_{j=1}^{9} cs_j)$. Obviously, a solution to the puzzle will be found when its cost is 0. See example of score calculations in page 293 of the article above.
   - Each neighbour of the current solution consists of a solution obtained by swapping the values of two non-fixed cell positions belonging to the same 3×3 subsquare. The neighbourhood search follows the simulated annealing paradigm - so first a nonfixed cell (cell that is not a given) is picked at random; next another non-fixed cell in the same square of the first cell is randomly chosen.

   (a) Give a pseudocode for the simulated annealing algorithm.

   (b) Implement your method, experimenting to find appropriate values for initial temperature t_0, cooling factor α (the author suggests $\alpha = 0.99$) and c_{max}. Note the stopping criteria should be when cost is 0 or c_{max} is reached, so c_{max} should be chosen to be very high and hopefully never invoked.

   (c) Test your method on the 49151 known Sudoku puzzles with 17 givens provided at:

   `http://mapleta.maths.uwa.edu.au/~gordon/sudoku17`

   Choose a few different design parameter combinations, and for each of them provide statistics on the solution of all the puzzles in the collection (average running time, average number of iterations, # of successful solutions).

2. (50 points) **SUDOKU by tabu search**

   Using a similar type of design as the previous method (neighbourhood, cost function, etc), modify the algorithm in part one to implement a Tabu search algorithm.

   (a) Give a pseudocode for the tabu search algorithm.

   (b) Implement your method, experimenting to find appropriate values for tabu lifetime L.

   (c) Test your method on the 49151 known Sudoku puzzles with 17 givens tested in part 1. Provide statistics on the solution of the puzzles (average running time, average number of iterations, # of successful solutions) for different choices of L, c_{max}.

   (d) Fix the best parameter choices for both algorithms (annealing and tabu) and compare the results, drawing some conclusions from your experiments.