CSCI 135 – Software Design and Analysis I
Assignment 1

Introduction

This assignment gives you practice programming with arrays and functions. As with all programs, you are free to discuss with or take inspiration from the instructor, other students and other sources, but your implementation must be your own work.

Your program must be written in C++ and must compile and run on eniac.geo.hunter.cuny.edu. Your program should also follow the guidelines set out here: http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci235/programming_rules.pdf. Submit the source code for your program via Blackboard.

Submission and Grading

Submit your source code (.cpp and .hpp files) for your implementation on Blackboard. Your program will be graded as follows:
55% = Correctness
25% = Design
10% = Style
10% = Documentation

Assignment

Write a program which solves sudoku puzzles. From Wikipedia (http://en.wikipedia.org/wiki/Sudoku): Sudoku is a logic-based, combinatorial number-placement puzzle. 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. The same number may not appear twice in the same 9x9 playing board row or column or in any of the nine 3x3 subregions of the 9x9 playing board. Below is an example of of a partially-filled Sudoku puzzle with each 3x3 subgrid shown in alternating colours.

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

Your program must do the following:

- Allow the user to enter a partially-filled sudoku grid, like the above
- Print the partially-filled sudoku grid with unknown portions indicated with a special character, such as a "?" character
- Attempt to solve the sudoku puzzle – see details below
- Print the solved sudoku puzzle – but see more below

On the attempt to solve the sudoku puzzle:

There are several techniques for solving sudoku puzzles. Many are based on keeping track of the "definite" numbers in the grids (one in each cell) along with a list of "candidates" – the number or numbers which could be placed in the cell. Your implementation only has to implement two techniques, specified below:

- Eliminate candidates based on presence in the row, column or 3x3 subgrid. For example, none of the empty squares in the topmost row may contain a "1" because of the presence of that number in the first row and fourth column. Likewise, none of the empty cells in the bottom-right subgrid may contain a "5" because that number already appears in the subgrid.
- Convert candidate to "definite" if it is the only remaining candidate in a cell.

On failure of the algorithm to find a solution and printing partially-solved puzzles:

Because your implementation will only implement the two techniques, there are several puzzles – usually the more difficult ones – which may will have no definite solution. Your implementation must detect when it stops making progress toward a solution, mark each cell with a non-numeric character and print the list of candidates for each cell.

Your implementation must use arrays, functions and local variables.
Your main may be like the one below:

```
int main () {
      int definite [9][9];
      bool candidate [9][9][9];

      enterDefinites(definite, candidate);
      print (definite, candidate, false);  // Do not print candidates; just a "?"
      solve (definite, candidate);
      print (definite, candidate, true);   // Print candidate list

      return 0;
}
```

Your program may operate like the below:

```
eniac $ ./sudokuSolver
Enter the starting state of the Sudoku puzzle
line by line with no spaces between each tile.
If a tile is empty, enter ?
???1?5?9?
14????67?
?8???24??
?63?7??1?
9????????3
?1??9?52?
??72???8?
?26????35
???4?9?6?
```

```
 -----------------------------
| ?  ?  ?  | 1  ?  5  | ?  9  ?  |
| 1  4  ?  | ?  ?  ?  | 6  7  ?  |
| ?  8  ?  | ?  ?  2  | 4  ?  ?  |
 -----------------------------
| ?  6  3  | ?  7  ?  | ?  1  ?  |
| 9  ?  ?  | ?  ?  ?  | ?  ?  3  |
| ?  1  ?  | ?  9  ?  | 5  2  ?  |
 -----------------------------
| ?  ?  7  | 2  ?  ?  | ?  8  ?  |
| ?  2  6  | ?  ?  ?  | ?  3  5  |
| ?  ?  ?  | 4  ?  9  | ?  6  ?  |
 -----------------------------
Adding all candidates to puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
No more candidates to eliminate.
 -----------------------------
| 6  7  2  | 1  4  5  | 3  9  8  |
| 1  4  5  | 9  8  3  | 6  7  2  |
| 3  8  9  | 7  6  2  | 4  5  1  |
 -----------------------------
| 2  6  3  | 5  7  4  | 8  1  9  |
| 9  5  8  | 6  2  1  | 7  4  3  |
| 7  1  4  | 3  9  8  | 5  2  6  |
 -----------------------------
| 5  9  7  | 2  3  6  | 1  8  4  |
| 4  2  6  | 8  1  7  | 9  3  5  |
| 8  3  1  | 4  5  9  | 2  6  7  |
 -----------------------------
Solve another? Y
Enter the starting state of the Sudoku puzzle
line by line with no spaces between each tile.
If a tile is empty, enter ?
5?7?42???
???9??6??
??6?5????
?1?8??76?
8???????1
?75??6?3?
????6?9??
??2??4???
???38?1?6
 -----------------------------
| 5  ?  7  | ?  4  2  | ?  ?  ?  |
| ?  ?  ?  | 9  ?  ?  | 6  ?  ?  |
| ?  ?  6  | ?  5  ?  | ?  ?  ?  |
 -----------------------------
| ?  1  ?  | 8  ?  ?  | 7  6  ?  |
| 8  ?  ?  | ?  ?  ?  | ?  ?  1  |
| ?  7  5  | ?  ?  6  | ?  3  ?  |
 -----------------------------
| ?  ?  ?  | ?  6  ?  | 9  ?  ?  |
| ?  ?  2  | ?  ?  4  | ?  ?  ?  |
| ?  ?  ?  | 3  8  ?  | 1  ?  6  |
 -----------------------------
Adding all candidates to puzzle.
```

```
Eliminating candidates from puzzle.
Eliminating candidates from puzzle.
No more candidates to eliminate.
 -----------------------------
| 5  A  7 | B  4  2 | C  D  E |
| F  G  H | 9  I  J | 6  K  L |
| M  N  6 | O  5  P | Q  R  S |
 -----------------------------
| T  1  U | 8  V  W | 7  6  X |
| 8  Y  Z | [  \  ] | ^  _  1 |
| `  7  5 | a  b  6 | c  3  d |
 -----------------------------
| e  f  g | h  6  i | 9  j  k |
| l  m  2 | n  o  4 | p  q  r |
| s  t  u | 3  8  v | 1  w  6 |
 -----------------------------
A: 3 8 9
B: 1 6
C: 3 8
D: 1 8 9
E: 3 8 9
F: 1 2 3 4
```

… etc.