CSI 5165 Combinatorial Algorithms — Fall 2005
Computer Science — University of Ottawa

Homework Assignment #1 (100 points, weight 15%)
Due: Wednesday, October 12, at 11:30 a.m. (in lecture)

---

1. (10 points) **Simple practice with combinatorial generation algorithms**
   Calculate the result for the following operations. Show your work.

   - Subsets:
     Give the SUCCESSOR and the RANK of 01010110 in the Gray code G^8.
   - k-subsets:
     Give RANK of $\{3, 6, 7, 9\}$ considered as a 4-subset of $\{0, 1, \ldots, 12\}$ in lexicographic and revolving-door order. What is the SUCCESSOR in each of these orders?
   - Permutations:
     Find the rank and successor of the permutation $[2, 4, 6, 7, 5, 3, 1]$ in lexicographic and Trotter-Johnson order.

     UNRANK the rank $r = 56$ as a permutation of $\{1, 2, 3, 4, 5\}$, using the lexicographic and Trotter-Johnson order.

2. (10 points) **Proving a recurrence for derangements**
   A *derangement* is a permutation $[\pi[1], \pi[2], \ldots, \pi[n]]$ of the set $\{1, 2, \ldots, n\}$ such that $\pi[i] \neq i$. Let D_n denote the number of derangements of $\{1, 2, \ldots, n\}$. Prove the following recurrence relation using induction on n:

$$\begin{aligned} D_1 &= 0, \\ D_2 &= 1, \\ D_n &= (n-1)\,(D_{n-1} + D_{n-2})\,, \quad \text{for } n \geq 3 \end{aligned}$$

3. (20 points) **Correctness of SUCCESSOR algorithm for Graycodes**
   Prove Theorem 2.2 of the textbook which states that Algorithm 2.3 (also given in class) correctly computes SUCCESSOR for the binary reflected Gray code. Prove it by induction on n.

4. (20 points) **Generating k-permutations** (Exercise 2.14)
   For a positive integer $k < n$, a k-permutation of $\{1, 2, \ldots, n\}$ is a list π of length k such that every element of the n-set occurs at most once in the list π. Develop a minimal change algorithm to generate all the k-permutations of $[1, n]$. At each step, this algorithm should change exactly one element.

   Hint: You may call algorithms (for other combinatorial structures) studied in class.

5. (40 points) **SUDOKU by backtracking**

SUDOKU is a placement puzzle in which symbols from 1 to 9 are placed in cells of a 9×9 grid made up of nine 3×3 subgrids, called regions. The grid is partially filed with some symbols (the "givens"). The grid must be completed so that each row, column and region contains exactly one instance of each symbol.

Example1: easy for humans

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

Example 2: medium for humans

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

Example 3: hard for humans

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

Write a **pseudocode** for a backtracking algorithm that solves **SUDOKU**.

Implement your algorithm and test for various instances (some instances will be specified at the course web page).

The input for your program consists of:

- m: the number of "givens" $(0 \leq m \leq 81)$
- m triples $(i_1, j_1, s_1), \ldots (i_m, j_m, s_m)$ representing the "givens" (If M is the SUDOKU matrix, then triple (i_k, j_k, s_k) signifies that $M[i_k, j_k] = s_k$).

The output of your program should consists of:

- the input grid;
- the solution grid;
- statistics on the algorithm performance such as: total number of basic steps (e.g. total number of symbol-to-cell placements, if your basic backtrack is on cells), running time (CPU time for the solution, not including I/O), etc. More statistics may be specified later.

Write your program in some high level programming language such as C, C++, Java. Hand in pseudocode, program and output results. Please, specify the platform you runned your tests on (machine speed, machine RAM and operating system).