CSI 5165 Combinatorial Algorithms — Winter 2009
Computer Science — University of Ottawa

Homework Assignment #1 (100 points, weight 15%)
Due: Thursday, February 5, at 8:50 a.m. (in lecture)
Deadline for e-mailing the test cases for exercise 4: Thursday January 22

Part 1: Generating elementary combinatorial objects

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. (30 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. You need to state and prove several facts, which will formalize the informal statements given in page 38 of the textbook:

   > Algorithm 2.3 works as follows. If $w(A)$ is even, then the last bit of A (namely a_0) is flipped; if $w(A)$ is odd, then we find the first "1" from the right, and flip the next bit (to the left). The last vector in G^n, which has no successor (added by editor: or sucessor $[0, \ldots, 0]$ in circular order) is $[1, 0, \ldots, 0]$ This corresponds to the set $\{1\}$.

   Hint: Prove the facts by induction on n.

3. (30 points) **Generating k-multisets of an n-set** (Exercise 2.13)
   A *multiset* is a set with (possibly) repeated elements. A k-multiset is one that contains k elements (counting repetitions). Thus, for example, $\{1, 2, 3, 1, 1, 3\}$ is a 6-multiset. The k-multisets of an n-set can be ordered lexicographically, by sorting the elements in each multiset in non-decreasing order and storing the result as a list of length k.

   When writing your algorithms, you will need to consider the following quantity:
   L_k^n = number of k-multisets of an n-set.

(a) (3 points) Give a recurrence formula for L_k^n.

(b) (2 points) Give a brief algorithm which computes and stores L_i^m, for all $i \leq k$ and $m \leq n$, on a table.

(c) (25) Develop successor, ranking and unranking algorithms for the k-multisets of an n-set. You may simply refer to the table values calculated in the previous part.

Hint: Try to adapt the algorithms for lexicographical ordering of k-subsets of an n-set. Note that you will use quantities L_i^m in place of quantities $\binom{p}{s}$.

Example: $k = 3$, $n = 4$, $L_3^4 = 20$

rank	T	$\vec{T}$
0	$\{1,1,1\}$	$[1,1,1]$
1	$\{1,1,2\}$	$[1,1,2]$
2	$\{1,1,3\}$	$[1,1,3]$
3	$\{1,1,4\}$	$[1,1,4]$
4	$\{1,2,2\}$	$[1,2,2]$
5	$\{1,2,3\}$	$[1,2,3]$
6	$\{1,2,4\}$	$[1,2,4]$
7	$\{1,3,3\}$	$[1,3,3]$
8	$\{1,3,4\}$	$[1,3,4]$
9	$\{1,4,4\}$	$[1,4,4]$
10	$\{2,2,2\}$	$[2,2,2]$
11	$\{2,2,3\}$	$[2,2,3]$
12	$\{2,2,4\}$	$[2,2,4]$
13	$\{2,3,3\}$	$[2,3,3]$
14	$\{2,3,4\}$	$[2,3,4]$
15	$\{2,4,4\}$	$[2,4,4]$
16	$\{3,3,3\}$	$[3,3,3]$
17	$\{3,3,4\}$	$[3,3,4]$
18	$\{3,4,4\}$	$[3,4,4]$
19	$\{4,4,4\}$	$[4,4,4]$

Part 2: Backtracking Algorithm Design and Implementation

4. (30 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 a 9×9 matrix representing the SUDOKU puzzle, where empty spaces in the grid are entered as 0's.

The input files must be in a standard format, so that all programs can share a pool of examples the class will create: each row must consist of 9 digits (no space) followed by a newline. Each student will type 3 input files corresponding to an easy, a medium and a hard SUDOKU, and email the plain text files to the instructor. Filenames should be FLe.txt, FLm.txt, FLh, where F and L are the first initial for your first and last name. E.g. I'd create LMe.txt, LMm.txt and LMh.txt The files will be posted in the course web page.

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.

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).