COSC 4P03 – Advanced Algorithms
Winter 2018
Assignment #2

Due Date: 19th March, 12:30pm Late Date: 22nd March, 12:30pm
This assignment accounts for 10% of your final grade and is worth a total of 100 marks.

In this assignment you are to use backtracking to solve a problem related to latin squares.
A latin square of order *n* is an *n*x*n* matrix in which:

- All entries are integers in the range [1..*n*],
- The same value cannot be repeated in any row, and
- The same value cannot be repeated in any column.

Note that another possible view is that each row is a permutation of [1..*n*] and each column is a permutation of [1..*n*].

We will consider two latin squares to be equivalent if one can be obtained from the other by a permutation of the rows and/or a permutation of the columns. Note: this definition of equivalence is for the purposes of this assignment – there are indeed further possibilities, however we shall not consider them. According to our given definition, the following are equivalent latin squares of order 4:

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

3	2	1	4
2	1	4	3
1	4	3	2
4	3	2	1

Meanwhile, the following latin square is not equivalent to either of the above:

1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

Question 1 (70 marks):
Write a program which uses a backtracking algorithm to compute the number of inequivalent latin squares for a given value of *n*, according to our above definition of equivalence You should run this program once for each different value of *n* from 3 upwards, up to whatever value your program can complete in no more than 1 day of CPU time on a single processor.
Input: *n* (integer)
Output: The number of inequivalent latin squares of order *n*, followed by a list of 4 inequivalent latin squares of order *n* (or as many as exist, if this is less than 4), and an indication of CPU time. Save your output for each run in a separate text file, and submit all of these files.
Important note: Marks will be allocated for efficiency – see recommendations below. **Ensure that you have adequate documentation to explain the decisions you have made to help efficiency.**

Question 2 (30 marks);
Write a program to estimate the number of nodes at each level of the backtracking search tree for your solution to question 1 above. Use this program to estimate the number of nodes at each level of the tree when used to compute the number of inequivalent latin squares of order *n* for all values of *n* between 9 and 15, inclusive. Your program should randomly choose at least 5 *different* paths for each one of these values, and average out their results to produce your estimate.
Input: *n* (integer)
Output: A table indicating, for each level of the search, the estimated number of nodes at that level. Note: if the estimated number is 0, then it is not necessary to print further levels as these will also be 0. Save your output for each run in a separate text file, and submit all of these files.

Recommendations:
Given a large enough value of *n*, your program from part 1 will take an extremely long time to run. I strongly recommend you do all of the following:

1. Try small values of *n* before moving on to the larger ones. You might be very surprised to discover the huge difference in CPU time to go from order *n* to order *n*+1.
2. Carefully track the progress of your program so that you will know how much of the search remains, allowing you to "kill" it if necessary.
3. Think about what constitutes a "level" of the search.
4. Think carefully about which candidates you actually need to test at each level of the search.
5. Think carefully about when and how to test for equivalence.
6. Think carefully about whether there are any assumptions you can make about the structure of the matrix.

Additional Notes:

1. Marks are allocated for producing the correct results while following the given instructions. Correct output with incorrect implementation and/or without appropriate documentation will result in a poor grade.
2. In order to simplify marking, try to adhere to the requested format as much as possible.

Submission Requirements:
All of the following must be placed in a sealed envelope in the 4P03 assignment box:

1. A cover sheet, available from `http://www.cosc.brocku.ca/forms/cover`, completely filled out. Your assignment will not be marked unless one is submitted with the assignment.
2. Commented and properly documented listings for all source code for your program.
3. Adequate documentation to explain (and show the validity of) the decisions you have made to improve efficiency.
4. Printouts of all output as specified above.
5. Any information required to run your programs.

You must also submit your assignment electronically so that it can be checked for plagiarism using `MOSS`. To do this, create a directory on Sandcastle containing all files for this assignment, and run the script `submit4p03` from this directory.