Programmming Assignment #1

Exercise A (20 points) Another way to order the subsets of an n-set is to order them first in increasing size, and then in lexicographic order for each fixed size. For example, when $n = 3$, this ordering for the subsets of $S = \{1, 2, 3\}$ is:

$$\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}.$$

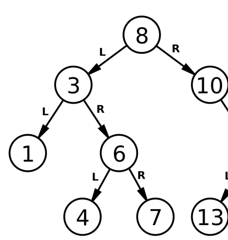
Define and implement unranking, ranking and successor algorithms for the subsets with respect to this ordering. Show the output of these functions for n-sets with n = 4, 5, and 6.

Exercise B (40 points) A *c-colored permutation* of an n-set is a permutation where each element is assigned one of c colors. For example, the 3-colored permutations of $\{1, 2, 3\}$ for the colors black, red and blue are listed below.

123	312	231	123	312	231	123	312	231
123	312	231	123	312	231	123	312	231
123	312	231	123	312	231	123	312	231
213	132	321	213	132	321	213	132	321
213	132	321	213	132	321	213	132	321
213	132	321	213	132	321	213	132	321
123	312	231	123	312	231	123	312	231
123	312	231	123	312	231	123	312	231
123	312	231	123	312	231	123	312	231
213	132	321	213	132	321	213	132	321
213	132	321	213	132	321	213	132	321
213	132	321	213	132	321	213	132	321
123	312	231	123	312	231	123	312	231
123	312	231	123	312	231	123	312	231
123	312	231	123	312	231	123	312	231
213	132	321	213	132	321	213	132	321
213	132	321	213	132	321	213	132	321
213	132	321	213	132	321	213	132	321

1. Define an order for 2-colored permutations of an n-set.
2. Using the ordering defined in (1), define ranking, unranking and successor functions for 2-colored permutations.
3. Using your successor function, implement a function that given n as input list all 2-colored permutations over an n-set, following the order defined in (1).
4. Show the output of the functions in (2) for values of n = 4, 5, and 6.

Exercise C (40 points) A *binary search tree* is a tree where each node has at most two children, each node has a key (for this exercise we will consider integers as keys, with their usual strict order $<$), and its left and right descendant have keys which are strictly smaller and larger, respectively. An example of a binary search tree is shown below.



Write a program that uses backtracking to generate all possible binary search trees for a given set of keys.

Show the output of your program for the following sets of keys: $\{-3, -2, 0, 2\}$, $\{1, 2, 5, 7, 8, 9\}$, and $\{-100, -50, -25, -5, 5, 25, 50, 100\}$. You will receive additional points, if you implement a way of showing your answer in a graphical way.

General Comments:

1. Read **carefully** all details about Programming Assignments in the Moodle page of the course.
2. Provide a **pseudocode** of your algorithms (with similar level of detail as the algorithms given in textbook). Also add any comments or extra **explanations** necessary to understand why your pseudocode works.
3. **Implement** the algorithms, providing a well documented **listing** of the code in C/C++ or Python (but do not include this listing in your report!). Include a README file (plain text) explaining how to run your code.
4. It should be possible to **easily run** your code with different values of the parameters, which are provided as input during the run (not hard-coded in your program).
5. It is useful to print intermediate results in some cases (e.g., during backtracking) with suitable information of what is been printed.
6. Start on all exercises early enough, so that there is time for questions. Some students may be less versed on algorithmic aspects, while others may need to understand more deeply some materials covered in class before applying them in a different context. Use our meetings on Thursday for these purposes!