Sélection internationale Session 2017
École normale supérieure Paris
Épreuve de culture scientifique - **Informatique** Duration : 3 hours

For candidates who chose **computer science** as **primary specialisation**
If you cannot answer a question, you may use it as hypothesis to later questions.
Calculators are not allowed.

Exercise 1. We suppose that we are given n identical-looking coins among which some are fake (or counterfeit). All genuine coins have exactly the same weight w_0 and all fake coins have exactly the same weight $w_1 < w_0$. We are given a two pan balance scale which given two arbitrary sets of coins (among these n coins) tell us if these two sets have exactly the same weight or which set is the lighter. Without the help of this balance, the coins are indistinguishable. We assume that there is at least one counterfeit coin among the n coins.

1. Propose an algorithm which identifies (at least) one fake coin when we are given $n = 8$ coins. This algorithm should make a minimum number of weighings (but you're not asked to prove that this number is minimal). How many weightings makes your algorithm in the worst case?

2. Propose an algorithm which identifies (at least) one fake coin in $\log_2(n)+O(1)$ weightings.

3. Propose an algorithm with $\log_2(n) + O(1)$ weightings that allows to divide the coins into three sets S_1, S_2 and S_3 such that

   - S_1 and S_2 contain the same number of genuine coins;
   - S_1 and S_2 contain the same number of fake coins;
   - S_3 contains at most two coins.

4. Deduce an algorithm with $\log_2(n)+O(1)$ weightings which gives the parity of the number of fake coins among the n coins.

5. Deduce an algorithm with $\log_2^2(n)+O(\log_2(n))$ weightings which gives the number of fake coins among the n coins.

6. Show (using a combinatorial argument) that an algorithm which gives the number of fake coins among the n coins has to make at least $\log_3(n)$ weightings.

Exercise 2. We suppose that we have n nuts and n bolts of different sizes. Each nut corresponds to unique bold (and reciprocally). The nuts and bolts are almost of the same size and it is not possible to say whether a nut is larger than another one and whether a bolt is larger than another one. However, if one wants to associate a nut to a bolt, the nut is either too large, too small or exactly of the same size. We can make nut-bolt comparison and each comparison gives one result among these three possibilities.

1. Propose an algorithm which given a nut finds the corresponding bolt. Give its complexity in the number of nut-bolt comparisons (as a function of n).

2. Propose a probabilistic algorithm inspired by Quicksort to associate each nut to its bolt.

3. Let $X_{i,j}$ be the random variable which is equal to 1 if during the execution of the algorithm the nut i is compared with the bolt j and $X_{i,j} = 0$ otherwise. Express as a function of the $X_{i,j}$'s the number $T(n)$ of nut-bolt comparisons made by the algorithm.

4. Give a formula for $\mathbb{E}(X_{i,j})$ the expectation of $X_{i,j}$.

5. Show that $\mathbb{E}(T(n)) = O(n \log n)$ (where $\mathbb{E}(T(n))$ is the expectation of $T(n)$).

Exercise 3. You have n homeworks to hand in but you are late. The i-th homework needs h_i working hours and has to be finished before time t_i, otherwise you will be penalized: if the homework is finished at time t'_i, the late penalty will be 0 if $t'_i \leq t_i$ and $t'_i - t_i$ for $t'_i > t_i$. The goal is to determine for each i the starting time s_i of the homework i, in order to minimize the total penalty. Of course, you can only work on one homework at the same time.

1. Give a polynomial-time algorithm which given $(h_1, t_1), (h_2, t_2), \ldots, (h_n, t_n)$ outputs the starting times $s_1, s_2, \ldots, s_n$.

2. Show that your algorithm is correct (i.e. that it indeed minimizes the total penalty).

3. What is the time complexity of your algorithm?

4. Eventually, you realize that you will not be able to hand in all homeworks and you have to abandon some of them. Choosing not to hand in homework i gives you a penalty p_i. Give an efficient algorithm which given $(h_1, p_1, t_1), (h_2, p_2, t_2), \ldots, (h_n, p_n, t_n)$, outputs $(b_1, s_1), (b_2, s_2), \ldots, (b_n, s_n)$, where b_i is a Boolean variable, true if if you hand in homework i and false otherwise, and s_i is defined only when b_i is true and in this case, is equal to the starting time of homework i.

Exercise 4. The goal of this exercise is to conceive data structures to store temperature data S. One temperature measure is of the form $(t; d)$ where t is a real number (that can be negative) and represents a temperature (in Celsius degree) and where d is an integer corresponding to the number of days since December, 31 1999 and the date of the measure (d is always positive). It is possible to have several measures on the same day (that may contain the same temperature but not necessarily, S is therefore a multi-set since it may contain several times a pair (t, d)) and to have none.

The data structure for S should provide the following operations:

- insert$(t; d)$ which adds to S the measure $(t; d)$;
- mean$(d_1; d_2)$ where $d_1 \leq d_2$ are two dates, which outputs the uniform mean of the temperatures of all measures (t, d) from S such that $d_1 \leq d \leq d_2$ (i.e. the sum of all these temperatures divided by the number of measures).

1. Show briefly how, with a naive data structure, one can obtain a $O(1)$ complexity for each insertion and a $O(n)$ complexity for each mean (where n is the cardinality of S).

2. Show how, with a different method, one can obtain a $O(D)$ complexity for each insertion and a $O(1)$ complexity for each mean (where D is the total number of days between December, 31 1999 and the present day).

3. Propose a data structure with $O(\log n)$ complexity for each insertion and a $O(\log n)$ complexity for each mean.