16 (100 pts.) **Piazza.**

Recently the ECE department discussed Piazza engagement. Haitham believes the student engagement in CS/ECE 374B is highly skewed, despite the many questions being asked. He claims that the top 5% of student askers (who ask many questions) contribute more than ten times the total number of questions asked by the bottom 90% of student askers (who ask little to no questions). To verify this claim, Andrew downloaded the Piazza statistics which give him an n element *unssorted* array A, where A[i] is the number of questions asked by student i.

16.A. (20 pts.) Describe an O(n)-time algorithm that given A checks whether the top 5% contribute more than ten times the questions asked by the bottom 90% together. Assume for simplicity that n is a multiple of 100 and that all numbers in A are distinct. Note that sorting A will easily solve the problem but will take Ω(n log n) time.

16.B. (70 pts.) More generally we may want to compute the total number of questions of the top p% of student askers for various values of p. Suppose we are given A and k distinct numbers 0 < p_1 < p_2 < ... < p_k < 100% and we wish to compute the total number of questions of the top p_i% of student askers for each 1 ≤ i ≤ k. Assume for simplicity that np_i is an integer for each i.

Describe an algorithm for this problem that runs in O(n log k) time. You should prove the correctness of the algorithm and its runtime complexity.

Note that sorting will allow you to solve the problem in O(n log n) time but when k ≪ n, O(n log k) is faster. Also, an O(nk) time algorithm is relatively easy by repeating the previous part k times.

16.C. (10 pts.) In an effort to encourage more discussion on Piazza, you will receive 10 points of credit if by March 23, 2020, you have asked a question or answered a question or contributed to any Piazza discussion at least once.

17 (100 pts.) **Strings**

Let Σ be a finite alphabet and let L_1 and L_2 be two languages over Σ. Assume you have access to a subroutine *IsStringInL*(u, L) which returns true if u ∈ L and false otherwise. Assume that *IsStringInL*(u, L) runs in constant time O(1).

Using the subroutine as black boxes describe an efficient algorithm that given an arbitrary string w ∈ Σ* decides whether w ∈ (L_1 • L_2)*. Evaluate the running time of your algorithm in terms of n = |w|. 
(100 pts.) **Invest.**

You have a group of investor friends who are looking at $n$ consecutive days of a given stock at some point in the past. The days are numbered. $i = 1, 2, \ldots, n$. For each day $i$, they have a price $p(i)$ per share for the stock on that day.

For certain (possibly large) values of $k$, they want to study what they call $k$-shot strategies. A $k$-shot strategy is a collection of $m$ pairs of days $(b_1, s_1), \ldots, (b_m, s_m)$, where $0 \leq m \leq k$ and

$$1 \leq b_1 < s_1 < b_2 < s_2 < \cdots < b_m < s_m \leq n.$$ 

We view these as a set of up to $k$ nonoverlapping intervals, during each of which the investors buy 1,000 shares of the stock (on day $b_i$) and then sell it (on day $s_i$). The return of a given $k$-shot strategy is simply the profit obtained from the $m$ buy-sell transactions, namely,

$$1000 \cdot \sum_{i=1}^{m} (p(s_i) - p(b_i)).$$

Design an efficient algorithm that determines, given the sequence of prices, the $k$-shot strategy with the maximum possible return. Since $k$ may be relatively large, your running time should be polynomial in both $n$ and $k$. 