1 (2). The input is an integer array a of n elements. It is required to find the number of inversions in a, i.e., the pairs of indices (i, j) such that $i < j$ and $a[i] > a[j]$. Construct an $O(n \log n)$ algorithm that finds the number of inversions in a.

Hint. Modify Merge sort

2 (3). RAM stores k arrays $A^1, A^2, \ldots, A^k$, each of which stores integers from 1 to n, and the sum of array lengths (total number of elements) is also equal to n. Build an algorithm that sorts all arrays in $O(n)$.

3 (3). Let the integer array $a[1], \ldots, a[n]$ be strictly unimodal to the maximum. It means that there exists t such that

$$a[1] < a[2] < \cdots < a[t] > a[t+1] > \cdots > a[n-1] > a[n], \quad 1 \leqslant t \leqslant n.$$

1. It is allowed to retrieve a single array's element per move (the input of the query is i the output is $a[i]$). Prove that it is possible to find the maximum element $a[t]$ in at most $O(\log n)$ moves.

4 (2). There are n coins, the one of which is fake, and a pan balance that can be used to find which of any two given coins is heavier. Real coins all have the same weight, while a fake coin is lighter. You can put any number of coins on each pan. Prove that the fake coin can be found in $\log_3 n + c$ weighings.

5 (3). Prove that under the conditions of the previous problem, finding the fake coin requires $\log_3 n + c$ weighings.

6 (4). There are two sorted arrays of length n of different elements. Propose $O(\log^2 n)$ algorithm for finding the median in the array consisting of all given $2n$ elements. Prove the correctness of the algorithm and estimate its complexity (number of comparisons). In this problem, retrieving an element is performed in O(1). Assume that both arrays are stored in RAM (so it is not needed to read the input).

7 (4). Determine whether the given number is the value of the given polynomial with natural (positive integer) coefficients at the natural point. Natural numbers $n, a_0, \ldots, a_n$, and y are the problem's input. It is necessary to determine whether there exists a natural number x such that

$$y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

Construct an $O(n \log y)$ algorithm that solves this problem. Assume that arithmetic operations cost $O(1)$.

8*. You have a 100-story building and a couple of marbles. You must identify the lowest floor for which a marble will break if you drop it from this floor. How fast can you find this floor if you are given an infinite supply of marbles? What if you have only two marbles? If the marble had not broken after the drop, it is as solid as before the drop, i.e. the number of flow had not changed.