Problem A. Going Down the Stairs

Idea: Илья Молочко
Development: Илья Молочко
Tutorial: Денис Ким

Expected difficulty: trivial
Success rate: 98.2 % (221/225 + 2 заморожено)

First to solve: 0:00:31 (BSUIR 2 : begunki)
Shortest solution: 32 байта (Belarusian NTU: Wrong Answer on Test 1)
Shortest jury solution 49 байт

Keywords: простая арифметика

There are exactly x steps between floors n and $n+1$, hence there are exactly x steps between any pair of adjacent floors. Going down from floor $n+1$ to floor 1 consists of n flights between adjacent floors, so the answer is $n \cdot x$.

The complexity is $\mathcal{O}(1)$.

Problem B. Metro

Idea: Илья Молочко
Development: Илья Молочко
Tutorial: Денис Ким

Expected difficulty: trivial
Success rate: 67.9 % (186/274 + 20 заморожено)

First to solve: 0:03:44 (Belarusian SU: static_cast<AC>)
Shortest solution: 133 байта (Belarusian NTU: Skill Issue)
Shortest jury solution 338 байт

Keywords: простая арифметика

All segments between stations i and $i+1$ for $1 \le i < n$ connect numbers of different parity, so each of them costs 1. The segment between stations n and 1 connects numbers of the same parity if and only if n is odd: in that case it costs 2, otherwise it costs 1.

Let $d = |s - f|$. One of the two arcs of the ring goes from the smaller number to the larger and does not pass through the segment between n and 1; it consists of exactly d segments of cost 1, so it costs d. The opposite arc contains $n - d$ segments and necessarily uses the segment between n and 1. Its cost is $n - d$ if n is even, and $n - d + 1$ if n is odd. The answer is the minimum of the costs of the two arcs.

The complexity is $\mathcal{O}(t)$ ($\mathcal{O}(1)$ per test case).

Problem C. Signs

Idea: Ефим Климашевский
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: trivial
Success rate: 82.6 % (194/235 + 5 заморожено)

First to solve: 0:01:38 (Belarusian SU: Team spilis)
Shortest solution: 171 байт (BSUIR: AgRE$ty)
Shortest jury solution 148 байт

Keywords: строки

A substring of length 3 is a palindrome if and only if its first and third characters are equal: the middle character may be anything. Therefore it is enough to check all windows $s_i s_{i+1} s_{i+2}$: if $s_i = s_{i+2}$ holds for at least one i, the answer is `YES`, otherwise `NO`.

The complexity is $\mathcal{O}(\sum |s|)$.

Problem D. Logo

Idea:	Алексей Толстиков
Development:	Андрей Лукашевич
Tutorial:	Денис Ким

Expected difficulty:	very easy
Success rate:	68.0 % (157/231 + 14 заморожено)

First to solve:	0:08:48 (Grodno SU #1: Bl9ShK4)
Shortest solution:	331 байт (Vitebsk SU: 1/3_Braincell)
Shortest jury solution	732 байта

Keywords: реализация, конструктив

To find the minimum m, round the square root of n up.

The $m \times m$ field can be filled, for example, as follows. Fill the $(m-1) \times (m-1)$ square in the top-left corner with characters '`#`' and place the remaining $k = n - (m-1)^2$ characters '`#`' in the L-shaped frame of $2m-1$ cells (the right column and the bottom row). Traverse the frame as a path from top to bottom along the right column, then from right to left along the bottom row (do not repeat the bottom-right cell) and fill the first k cells of the path. The empty cells form a suffix of this path and are therefore connected; when $n = m^2$ the whole table is filled.

The complexity is $\mathcal{O}(n)$.

Problem E. Magical Array

Idea:	Илья Молочко
Development:	Илья Молочко
Tutorial:	Денис Ким

Expected difficulty:	very easy
Success rate:	27.5 % (112/408 + 87 заморожено)

First to solve:	0:08:38 (Belarusian SU: Team spilis)
Shortest solution:	292 байта (Brest STU: Borisevich's theorem)
Shortest jury solution	478 байт

Keywords: инвариант, сортировки

Adjacent elements can be swapped if and only if they have different parities. When swapping, an odd number "jumps over" an even one. The invariant of this operation is the relative order of the odd elements and the relative order of the even elements. Conversely, any permutation that preserves the invariant can be obtained by such swaps.

The array can be made non-decreasing if and only if the odd elements are already in non-decreasing order and the even ones are too: any violation of this order is part of the invariant. This can be checked in a single pass over the array.

The complexity is $\mathcal{O}(\sum n)$.

Problem F. Message Filter

Idea: Иван Лукьянов
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: very easy
Success rate: 37.6 % (59/157 + 45 заморожено)

First to solve: 0:20:36 (Belarusian SU: Team spilis)
Shortest solution: 989 байт (Belarusian SU: 128 burbalka fans)
Shortest jury solution 1137 байт

Keywords: строки, реализация, перебор

Laughter is determined by a vowel v, a consonant c, and a type: whether it starts with a vowel or with a consonant. There are 5 vowels, 21 consonants, and 2 types, hence $5 \cdot 21 \cdot 2$ candidates. For fixed letters and type, a longest alternating subsequence can be built greedily from left to right: each time take the leftmost occurrence of the required letter to the right of those already taken. Deferring an occurrence is never better — a more left position leaves more freedom on the right. If the obtained length is even, drop the last character so that the laughter starts and ends with the same letter. The complexity of this solution is $\mathcal{O}(|\Sigma|^2 \cdot |s|)$, where $|\Sigma|$ is the size of the alphabet used.

It is possible (but not necessary) to solve the problem faster. A candidate string of length ℓ is uniquely determined by the first letter, the second letter, and ℓ. Therefore, among candidates of positive length, one chooses the greatest length, and on equal lengths — the smaller first letter and, if $\ell > 1$, the smaller second letter. All pairs can be maintained simultaneously in one pass over s: when a character is read, the lengths of the pairs it belongs to are updated. The complexity is $\mathcal{O}(|\Sigma| \cdot |s|)$.

Problem G. Multiple and Divisor

Idea: Иван Лукьянов
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: easy
Success rate: 29.9 % (81/271 + 91 заморожено)

First to solve: 0:08:06 (BSUIR: Prime limit)
Shortest solution: 493 байта (Grodno SU #12: RMStar)
Shortest jury solution 363 байта

Keywords: теория чисел, сортировки

Sort the array: $a_1 \le a_2 \le \ldots \le a_{2n}$. A number that divides x is at most x, and a multiple of x is at least x, so $a_n \le x \le a_{n+1}$. Therefore, the n smallest numbers must divide x, and x must divide the n largest ones. Thus $L = \operatorname{lcm}(a_1, \ldots, a_n)$ divides the answer, and the answer divides $G = \gcd(a_{n+1}, \ldots, a_{2n})$. A suitable x exists if and only if L divides G; in that case, any x with $L \mid x \mid G$ works.

One may take G as the answer: it always divides the upper half, and if some number of the lower half does not divide G, then no divisor of G works either. L also works, but when computing it one must watch for overflow: if the LCM becomes greater than 10^9, there is no answer.

The complexity is $\mathcal{O}(n \log n)$.

Problem H. Guess the Letter

Idea: Иван Лукьянов
Development: Денис Ким
Tutorial: Денис Ким

Expected difficulty: easy
Success rate: 32.7 % (86/263 + 61 заморожено)

First to solve: 0:19:17 (Belarusian SU: simply lovely)
Shortest solution: 316 байт (Gomel SU: C++_++))
Shortest jury solution 428 байт

Keywords: двоичный поиск

We keep the set of letters that may still have been chosen. Initially, this is all 26 lowercase English letters.

Each query consists of the current set of candidates. If their number is odd, we append one letter that is no longer a candidate so that the length becomes even. The chosen letter occurs in the query, so the jury must delete exactly one of its occurrences. Therefore, this letter is absent from the answer, and all candidates that returned can be crossed out. Thus, the set shrinks by about half.

While there is more than one candidate, we repeat the query. This takes $\lceil \log_2 |\Sigma| \rceil$ queries, where $|\Sigma|$ is the size of the alphabet. When one letter remains, we print it.

The complexity of the program itself is $\mathcal{O}(|\Sigma|)$.

Problem I. Likes

Idea: Алексей Толстиков
Development: Андрей Лукашевич
Tutorial: Денис Ким

Expected difficulty: easy-medium
Success rate: 37.8 % (59/156 + 68 заморожено)

First to solve: 0:22:46 (Belarusian SU: It's all decided. Accept it and relax!)
Shortest solution: 614 байт (Belarusian SU: static_cast<AC>)
Shortest jury solution 439 байт

Keywords: два указателя, префиксные суммы

Let $d_0 = 0$ and $s_0 = 0$. For each $j = 1, \ldots, n$, the segment of days $d_{j-1} + 1, \ldots, d_j$ contains exactly $s_j - s_{j-1}$ likes in total, and they may be distributed arbitrarily among the days of this segment as nonnegative integers.

To maximize the sum on a single window of length T, it is optimal to place all likes of a segment onto one of its days. Then the window takes the entire mass of every segment it intersects and nothing from the others. The answer is the maximum of $s_j - s_{i-1}$ over pairs $i \le j$ for which there exists a window of length T that intersects both the i-th and the j-th segments (then it also intersects all segments in between).

Such a window exists if and only if $i = j$ or $T \ge d_{j-1} - d_i + 2$: it is enough to cover day d_i and day $d_{j-1}+1$, and this segment of length $d_{j-1} - d_i + 2$ can always be extended to a window of length T inside the range of days $1, \ldots, d_n$. Prefix sums are non-decreasing, so for a fixed j it is enough to take the smallest suitable i: the farther left the left endpoint, the larger the difference $s_j - s_{i-1}$. As j increases, this minimal i is non-decreasing. All best pairs can be found by the two pointers method: the pointers i and j correspond to the ends of the considered set of segments and move simultaneously from left to right — j iterates over the right end, and i is shifted right to the smallest index for which the pair (i, j) admits a window of length T.

The complexity is $\mathcal{O}(n)$.

The problem can be solved differently: iterate over all window starts using a difference array. The segment of days $d_{j-1} + 1, \ldots, d_j$ intersects a window with start x if and only if $d_{j-1} - T + 2 \le x \le d_j$. Add the mass of the segment onto this range clipped to $[1, d_n - T + 1]$, and take the maximum of the prefix sums. This solution runs in $\mathcal{O}(n + d_n)$ time, and the memory consumption is $\mathcal{O}(n + d_n)$.

Problem J. Antimatter

Idea: Андрей Лукашевич
Development: Андрей Лукашевич
Tutorial: Денис Ким

Expected difficulty: easy-medium
Success rate: 51.4 % (19/37 + 30 заморожено)

First to solve: 1:26:42 (Belarusian SU: simply lovely)
Shortest solution: 716 байт (Belarusian SU: Rizz godz)
Shortest jury solution 538 байт

Keywords: биты, префиксные суммы

For $A = 0, 1, 2, 3$ the optimal segments are $[0, 0]$, $[0, 1]$, $[2, 2]$, $[0, 2]$ respectively. From now on, $A \geq 4$. Let B be the highest bit of A ($B \geq 4$).

Let $s(n) = 0 \oplus 1 \oplus \ldots \oplus n$ for $n \geq 0$ and $s(-1) = 0$. We will look for a half-interval $(L, R]$ ($L \geq -1$), that is, the numbers from $L + 1$ to R. Then

$$(L + 1) \oplus \ldots \oplus R = s(R) \oplus s(L),$$

and at the end we add 1 to L. Clearly $(A - 1, A]$ satisfies the statement, but we will try to improve this answer.

Since $(4k) \oplus (4k + 1) \oplus (4k + 2) \oplus (4k + 3) = 0$, the value of $s(n)$ is determined by n modulo 4:

$n \bmod 4$	$s(n)$	$s(n) \bmod 4$
0	n	0
1	1	1
2	$n + 1$	3
3	0	0

Note that the high bits of n appear in $s(n)$ if and only if n is even.

Consider the remainders of L and R modulo 4 and write down whose high bits are taken into the result (at the top of the cell) and what the two low bits of the answer are (at the bottom of the cell).

$L \setminus R$	0	1	2	3
0	L, R 0	L 1	L, R 3	L 0
1	R 1	— 0	R 2	— 1
2	L, R 3	L 2	L, R 0	L 3
3	R 0	— 1	R 3	— 0

The best option for us is the one where high bits are taken from both L and R (this is possible only when $A \equiv 0 \pmod 4$ and when $A \equiv 3 \pmod 4$). Then (ignoring the two low bits) we can set $R = B$ and put the remaining high bits of A into L.

If high bits are taken only from L or only from R, then the high bits of R cannot be smaller, taken together, than the high bits of A. This is the best we can achieve if A modulo 4 is 1 or 2.

If high bits are taken from neither L nor R, then we can only obtain a number not exceeding 3, which does not work for us.

More formally, using the data from the table above, the optimal answer for different remainders of A is as follows:

- $A \equiv 0 \pmod 4$: $R = B$, $L = A \oplus B$.
- $A \equiv 1 \pmod 4$: $R = A - 1$, $L = 1$.
- $A \equiv 2 \pmod 4$: $R = A$, $L = A - 1$ (the trivial solution).
- $A \equiv 3 \pmod 4$: $R = B$, $L = (A \oplus B \oplus 3) + 2$.

At the end, do not forget to add 1 to L.

The complexity is $\mathcal{O}(t)$ ($\mathcal{O}(1)$ per test case).

Problem K. Art Gallery

Idea:	Андрей Лукашевич
Development:	Андрей Лукашевич
Tutorial:	Денис Ким

Expected difficulty:	medium
Success rate:	59.1 % (26/44 + 17 заморожено)

First to solve:	1:10:32 (Belarusian SU: simply lovely)
Shortest solution:	1523 байта (Belarusian SU: Dance, horse)
Shortest jury solution	1738 байт

Keywords: структуры данных, сет

After cropping, painting (i, j) has size $r_i \times c_j$, where $1 \le r_i \le h_{i,j}$ and $1 \le c_j \le w_{i,j}$ for all i, j. The total area of scraps is $S - \left(\sum_i r_i\right)\left(\sum_j c_j\right)$, where $S = \sum_{i,j} h_{i,j} w_{i,j}$. All r_i and c_j are positive, and the constraints on rows and columns are independent, so the product of the sums is maximized by taking the largest feasible values: $r_i = \min_j h_{i,j}$ and $c_j = \min_i w_{i,j}$.

It is enough to maintain the sum of areas S, the sum of height minima over rows, and the sum of width minima over columns. On a replacement in cell (x, y), only row x and column y change. To update the minima, keep an ordered structure in each row and each column: in C++ this is a `std::multiset` (or a `std::set` of pairs), in Java a `TreeMap` as a multiset (or `TreeSet`). The same value may occur several times, so you must erase exactly one instance. The time complexity is $\mathcal{O}((nm+k)\log(n+m))$, the memory complexity is $\mathcal{O}(nm)$.

Instead of multisets, one can store an array of heights in row order and an array of widths in column order in segment trees: the minima of the required row and column become queries on contiguous segments. The time complexity is $\mathcal{O}(nm + k\log(nm))$, the memory complexity is $\mathcal{O}(nm)$.

In all solutions, do not forget to use a 64-bit data type.

Problem L. Candidates Tournament

Idea:	Ефим Климашевский
Development:	Ефим Климашевский, Денис Ким
Tutorial:	Денис Ким

Expected difficulty:	medium
Success rate:	50.0 % (2/4 + 7 заморожено)

First to solve:	1:56:29 (* Guest team: BSUIR2)
Shortest solution:	3505 байт (* Guest team: BSUIR2)
Shortest jury solution	2548 байт

Keywords: динамическое программирование по дереву, вероятности

For each candidate i we store the probability p_i that he has won his current subtree (at a leaf $p_i = 1$). When a left subtree and a right subtree meet, the action mix of the winner of the right subtree equals

$\sum p_j \cdot s_j$ over the candidates on the right, where s_j is that candidate's distribution over rock, scissors, and paper. The win probability is linear in the opponent's strategy, so it is enough for candidate i on the left to play against this three-dimensional mix: we multiply p_i by the probability that s_i beats it. We update the right side symmetrically. Because of the balance condition, each candidate lies in $\mathcal{O}(\log n)$ subtrees, and the total time is $\mathcal{O}(n \log n)$.

At the root, the champion's action mix is again $\sum p_i \cdot s_i$. If Svetozar shows rock, he wins with probability $S + \frac{R}{2}$, where R and S are the champion's probabilities of rock and scissors; the formulas for scissors and paper are obtained by a cyclic shift.

The complexity is $\mathcal{O}(n \log n)$.

It may seem natural to store at a vertex only the action distribution of the subtree winner. If the two children have distributions A and B, the left one wins with probability

$$w = A_R B_S + A_S B_P + A_P B_R + \tfrac{1}{2}(A_R B_R + A_S B_S + A_P B_P),$$

and then both can be replaced by the mix $w \cdot A + (1 - w) \cdot B$. This yields a single "virtual" candidate with a mixed strategy. This is **wrong**: the winner is a specific person, and in the next match different candidates from the mix have different chances against the new opponent. The overall action mix cannot be substituted for the person.

Problem M. Binary Array and Sums

Idea:	Андрей Лукашевич
Development:	Андрей Лукашевич
Tutorial:	Денис Ким

Expected difficulty:	medium-hard
Success rate:	11.1 % (1/9 + 6 заморожено)

First to solve:	3:50:39 (Belarusian SU: Team spilis)
Shortest solution:	2473 байта (Belarusian SU: Team spilis)
Shortest jury solution	2932 байта

Keywords: графы, разностные массивы

Let $p_0 = 0$ and $p_i = a_1 + \ldots + a_i$. Then $a_i = p_i - p_{i-1}$ and $0 \le p_i - p_{i-1} \le 1$, and a query (l, r, s) is equivalent to the equality $p_r - p_{l-1} = s$.

We maintain the equalities on prefix sums with a disjoint-set union with offsets: in one class, the values p_i differ by a known constant. If the next equality contradicts those already known, there is no answer.

After that, the constraints $0 \le p_i - p_{i-1} \le 1$ remain. If the indices i and $i - 1$ lie in the same class, the difference is already fixed, and it is enough to check that it equals 0 or 1. If they lie in different classes, we obtain difference constraints on the values of the roots of the form $x_u - x_v \le w$.

Such a system is feasible if and only if the graph with an edge $v \to u$ of weight w has no negative cycle. The cycle is found by the Bellman-Ford algorithm; the distances give some feasible assignment of the roots. From it, the values p_i and the bits a_i are recovered. If a cycle is found, the array cannot be reconstructed.

The complexity is $\mathcal{O}(n^2 + m)$, since after contracting the equalities the graph has $\mathcal{O}(n)$ vertices and $\mathcal{O}(n)$ edges, while the equalities themselves are processed in $\mathcal{O}(m)$.

Problem N. Gnomes

Idea:	Егор Дубовик
Development:	Денис Ким
Tutorial:	Денис Ким

Expected difficulty:	medium-hard
Success rate:	8.8 % (15/171 + 167 заморожено)

First to solve:	0:55:13 (Belarusian SU: simply lovely)
Shortest solution:	2156 байт (Belarusian SU: simply lovely)
Shortest jury solution	2469 байт

Keywords: структуры данных, дерево отрезков

If the value x is already known, the number of its occurrences on $[l, r]$ under assignments can be counted in $\mathcal{O}(\log n)$: with a Fenwick tree over positions (+1 at those indices where x stands) or with an ordered set of those positions. In C++ one can use `ordered_set` from GNU pbds (`tree_order_statistics_node_update`), in Java, `TreeSet` or `TreeMap`. When assigning a_p, position p is removed from the set of the old value and inserted into the set of the new one. The memory consumption is $\mathcal{O}(n \log n)$.

There is no need to iterate over all distinct numbers on the segment. It is enough to obtain a small set of candidates for x that must contain the answer if it exists, and check each of them by counting the frequency.

Candidates for x can be obtained in several ways.

- **Voting with one counter.** Walk through the array and store the current value y and an integer c. The next element equal to y increases c; otherwise c decreases, and when $c = 0$ the new element is written into y and c becomes 1. If some value occurred strictly more than half the time, it will remain in y: each foreign element decreases its counter by at most one. The result does not depend on the order, so the same pair (y, c) is stored in the nodes of a segment tree and merged in $\mathcal{O}(1)$ (if the values coincide, the counters are added; otherwise the smaller is subtracted from the larger). A query on $[l, r]$ yields one candidate for x; its frequency is checked by counting. For the threshold "strictly more than half" this is enough: there is at most one such value.
- **Voting with two counters.** Store at most two pairs "value, counter". A matching value increases its counter; a new one occupies a free slot, and if there are no free slots, all counters are decreased by one (zeros are dropped). A value that occurs in more than a third of the positions cannot be fully displaced. A frequency greater than 1/2 implies a frequency greater than 1/3, so the answer, if it exists, lies among at most two candidates for x. The same pairs are merged in a segment tree in $\mathcal{O}(1)$ and checked by counting the frequency.
- **Median.** In a sorted subsegment, a value with frequency greater than 1/2 occupies the median position. The k-th order statistic on a segment with point updates is maintained by a segment tree; the found value is a candidate for x, and it is checked by counting the frequency. This structure works in $\mathcal{O}((n + q) \log^2 n)$ and **does not pass** the time limit.
- **Random sampling.** There is at most one value with frequency greater than 1/2 on a segment. It is enough to take several random positions: it will appear in the sample with constant probability. All distinct chosen numbers (there are few of them) are candidates for x, and they are checked by the same counting. Such solutions might fail some tests in some cases.

More straightforward solutions work in $\mathcal{O}((n + q) \log^2 n)$: each node of a segment tree stores a frequency map, or two nested Fenwick trees are used. This is too slow.

The complexity of voting and random sampling is $\mathcal{O}((n + q) \log n)$.