ТУЙМААДА XXVIII Международная олимпиада
TUYMAADA The 28th International Olympiad

PROBLEMS AND SOLUTIONS

INFORMATICS

ИНФОРМАТИКА

ЗАДАЧИ И РЕШЕНИЯ

Yakutsk 2021
Якутск 2021

XXVIII Международная олимпиада школьников по математике, физике и информатике «Туймаада». ИНФОРМАТИКА. – Якутск, 2021.

Сборник содержит задачи XXVIII Международной олимпиады «Туймаада» по информатике, а также возможные варианты решений. Олимпиада проводилась 24 июля – 3 августа 2021 года в г. Якутске. Олимпиада прошла в дистанционном формате, в каждый из двух соревновательных дней участникам было предложено решить по пять задач.

This booklet contains the problems and possible solutions of the 28th Tuymaada International Olympiad in informatics. The olympiad wash held on July 24 – August 3, 2021 in Yakutsk, Russia. The olympiad was held remotely, participants were invited to solve five problems on each of the two competition days.

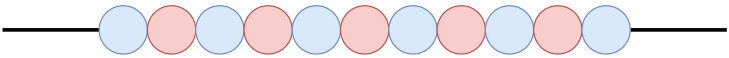
АВТОРЫ		*AUTHORS*
Юрий Саввич АНТОНОВ Северо-Восточный федеральный университет им. М. К. Аммосова (СВФУ)	A	*Yuri ANTONOV* *M. K. Ammosov North-Eastern Federal University (NEFU)*
Владимир Семенович Леверьев СВФУ	B	*Vladimir LEVERYEV* *NEFU*
Василий Алквиадович БУЛАТОВ Яндекс Такси	C	*Vasily BULATOV* *Yandex Taxi*
Александру ПЕТРЕСКУ студент, Оксфордский университет	D	*Alexandru PETRESCU* *student, Oxford University*
Семен Леонидович МЕКУМЯНОВ студент, НИУ ИТМО	E	*Semyon MEKUMYANOV* *student, ITMO University*
Владимир Семенович ЛЕВЕРЬЕВ	F	*Vladimir LEVERYEV*
Сергей Геннадьевич ВОЛЧЁНКОВ Ярославский государственный университет им. П. Г.Демидова, Ярославль	G	*Sergei VOLCHENKOV* *P. G. Demidov Yaroslavl State University, Yaroslavl*
Виталий Витальевич ИВАНОВ Яндекс.Маркет	H	*Vitalii IVANOV* *Yandex.Market*
Валерий Валерьевич ГАВРИЛЬЕВ Яндекс Карты	I	*Valery GAVRILIEV* *Yandex Maps*
Артем Тарасович ВАСИЛЬЕВ НИУ ИТМО	J	Artem VASILEV *ITMO University*

Problem statements

A. Necklace

Time limit: 1 second
Memory limit: 1024 MiB

The necklace is made up of $2N$ red and blue beads, so that among any K beads counting from its left edge, the number of the red beads is less or equal to the number of blue ones.



How many different necklaces can be made under these conditions?

Input
Positive integer number N ($N \leq 32$).

Output
Positive integer number, the answer.

Note
This problem contains three subtasks. Points for each test in each subtask are awarded independently.

Subtask 1 (points: 20)
$N < 10$.

Subtask 2 (points: 20)
$N < 20$.

Subtask 3 (points: 60)
$N \leqslant 32$.

Examples

standard input	standard output
1	2
2	6

B. Orchestra

Time limit: 1 second
Memory limit: 256 MiB

The conductor of a symphony orchestra once fell ill. Without the conductor, the musicians found it difficult to play in the common rhythm. To achieve this, some musicians look at one of their colleagues and try to play in sync with them. Other musicians just close their eyes and play by ear.

If you trace who is looking at whom, you can construct chains of musicians where each musician looks at the next person. Sometimes these chains end up in a loop. Let's assume that a musician cannot appear twice in the same chain.

Unfortunately, our orchestra has not rehearsed much so far and often starts to play out of sync. This happens because there are separate groups of musicians who only look at each other and don't look at musicians from other groups.

To learn how to play better, the musicians ask you to write a computer program to analyze the current situation.

Input

The first line of the input data contains N — the number of musicians in the orchestra ($N < 4 \cdot 10^4$).

Each of the following N lines contains one colleague's number who the current musician is looking at. If the number 0 appears, then the current musician doesn't look at anyone.

Output

In the first line, your program must output the number of independent groups of musicians K.

In the next K lines, your program must output the number of musicians in the maximal-length chain in each group of musicians. These numbers must appear in ascending order.

Note

The points for each test are awarded independently.

Examples

standard input	standard output
3 0 1 1	1 2
6 2 3 2 5 6 5	2 3 3

C. Find and prevent

Time limit: 5 seconds
Memory limit: 256 MiB

The Ministry of Trasportation of Treeland found out about a forthcoming illegal protest action in one of the cities of Treeland. Members of a group that calls itself "Biconnected" are threatening to organize a traffic nightmare by blocking ways in and out of one of the cities to draw attention to drawbacks of the current road network. It is known that their goal is to cause maximum damage to transport connectivity of the whole country.

There are N cities in Treeland and, as you may have guessed, the road network can be represented as a tree graph. Population of ith city is a_i people. If there are several cities that rebels can choose, then they are going to target the one with the smallest number.

The Ministry was recently a victim of severe budget cuts, therefore there is no one able to find out the target city of the activists, so they are asking for your help.

Transport connectivity of the country is equal to the number of pairs of people that can reach each other by roads or are living in the same city.

Input

The first line contains a single integer N ($1 \leq N \leq 1000000$).

The second line contains N numbers $a_1, a_2, \ldots, a_N$ ($1 \leq a_i \leq 100$).

Each of the following $N - 1$ lines describing the road system of Treeland contains 2 integers - pair of cities that are connected by a road.

Output

In the first line, output the number of the city in which the action will take place.

In the second line, print the new value of transport connectivity in the case that the action succeeds.

Note

This problem has three subtasks. Points for each are awarded only if the solution passes all the tests from this subtask.

Subtask 1 (points: 20)
$n \leq 1000$.

Subtask 2 (points: 60)
$n \leq 100000$.

Subtask 3 (points: 20)
$n \leq 1000000$.

Example

standard input	standard output
3 1 1 1 1 2 2 3	2 0

D. Expected xor

Time limit: 1 second
Memory limit: 256 MiB

There's a legend about a holy tree in Romania, with a number M written around its roots. Under the tree, an old man with a giant book is sitting from time immemorial, rolling N distinct magical dice. Each dice has A_i sides (A_i and M are coprime, i.e. the greatest common divisor of A_i and M is 1) numbered

from 0 to $A_i - 1$. When the dice are rolled, the old man takes the numbers on each dice b_i and records the value of

$$b_1 \textbf{ xor } b_2 \textbf{ xor } ... \textbf{ xor } b_N$$

in his book, where **xor** is bitwise exclusive OR. According to the legend, the combinations of the magical dice never repeat, and when they are exhausted, the world will end. Compute the arithmetic mean of the numbers in the old man's book at world's end modulo M, as described below.

Input
The first line contains positive integers M, N ($N \leq 50000$, $2 \leq M \leq 10^9 + 7$). The second line contains N positive integers A_i ($A_i \leq 2^{62}$, $i = 1, ..., N$).

Output
Print non-negative integer X ($X < M$) such that, if the answer is the fraction $\frac{U}{V}$ (where U and V are integers), then the product of X and V has the same remainder as U when divided by M:

$$X \cdot V \equiv U \pmod{M}.$$

Note
This problem has five subtasks. Points for each subtask are awarded only if the solution passes all the tests of that subtask.

Subtask 1 (points: 20)
$N \leq 5, M = 11, A_i < 2^3$.

Subtask 2 (points: 20)
$N \leq 100, M = 997, A_i \leq 2^5$.

Subtask 3 (points: 20)
$N \leq 50000, M = 10^9 + 7, A_i \leq 2^{62}$.

Subtask 4 (points: 20)
$N \leq 1000, M \leq 1000, A_i < 2^{30}$.

Subtask 5 (points: 20)
$N \leq 50000, M \leq 10^9 + 7, A_i \leq 2^{62}$.

Examples

standard input	standard output
11 1 10	10
10 3 7 9 3	8

Note

In the first example, the answer is $\frac{9}{2}$, so we print 10, because

$$10 \cdot 2 = 20 \equiv 9 \pmod{11}.$$

In the second example, the answer is $\frac{274}{63}$, so we print 8, because

$$8 \cdot 63 = 504 \equiv 4 \equiv 274 \pmod{10}.$$

E. Odd Taxi

Time limit: 2 seconds
Memory limit: 256 MiB

You have the task to take all the participants of the olympiad home. The city's roads can be represented as a rooted tree on n numbered vertices. The location of the Olympiad is in the vertex number 1. Each vertex number $i > 1$ is connected by a road to the vertex p_i ($p_i < i$). You have a contract with a company "`Odd Taxi`", which provides two types of cabs:

1. One-passenger, the fare is $dist_v \times x$ rubles, where $dist_v$ — the number of roads from the root 1 to destination v.
2. Two-passenger, the fare is $dist_v \times y$ rubles, and the cab driver takes one passenger to the destination vertex v and can drop off the second passenger at any vertex on the way to v including v.

The cab always takes the shortest path, a two-passenger cab cannot take only one passenger.

If each vertex i has a_i participants living there, what minimum amount of money should be spent to get everyone home?

Input
The first line contains the number of vertices n ($2 \leq n \leq 5000$), the fares of the first and second cab type, x and y, respectively ($1 \leq x \leq 10^5$, $1 \leq y \leq \min(2x-1, 10^5)$).

The second line contains $n-1$ numbers $a_2, \ldots, a_n$ — the number of participants living in each vertex ($0 \leq a_i \leq 10^5$).

The third line contains $n-1$ numbers $p_2, \ldots, p_n$ — parents of the vertices ($1 \leq p_i < i$).

Output
Output a single number — the amount of money required.

Note
This problem has three subtasks. Points for each subtask are awarded only if the solution passes all the tests of that subtask and all previous subtasks.

Subtask 1 (points: 20)
$n, a_i \leq 50$.

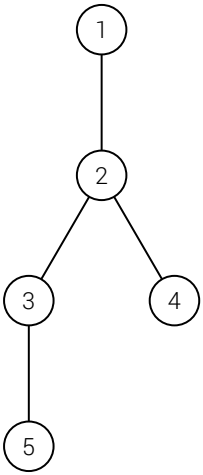
Subtask 2 (points: 30)
$n \leq 500$.

Subtask 3 (points: 50)
No additional constraints.

Examples

standard input	standard output
5 2 3 2 0 1 1 1 2 3 3	15
5 3 2 1 0 1 1 1 2 2 3	12

Note



The tree from the second sample. It's optimal to order two-passenger cab for the participants from vertices 2 and 5, one-passenger cab for the participant from vertex 4. In total we will pay $3 * y + 2 * x = 12$.

F. Alien music

Time limit: 1 second
Memory limit: 256 MiB

In the future, humanity has found a way to quickly travel across the galaxy. Alice is a young researcher from the 22nd century who studies the alien civilization of Rumatas.

In this civilization there is an analogue of earthly music. Alice found out that alien music is composed of N consecutive notes. Unlike earth music (which has only 7 different notes), an alien composer can choose from M_i options ($1 \leq i \leq N$) for the i-th note in the melody.

Also, it is important to know that each note has a tone, and the melody as a whole has an accent tone. It is required that the melody contains exactly 2 notes in an accent tone, to be considered harmonious.

Write a program to calculate the amount of different melodies that comply with the rule of harmony of alien music.

Input

The first line of the input data contains a natural number $N \leq 15$ and an accent tone number K ($0 < K \leq 20$).

The second line contains N natural numbers $M_i \leq 10$ — the number of different notes in the melody.

The following N lines contain M_i numbers, which indicate the tonality of each note.

Output

Write down the number of different alien melodies comply with the rule of harmony.

Note

The points for each test are awarded independently.

Example

standard input	standard output
3 1 4 5 2 1 2 3 4 1 2 3 3 4 1 2	8

Note

There will be 3 notes in the melody. The emphasis is on tonality #1.

For the first note there are 4 options, For the second note there are 5 different options (2 of them are of the same tone) and 2 options for the third note.

If the composer chooses the first and second notes for the accent tone, then there is only one option for the third note.

If the composer chooses the first and third notes for the accent tone, then there are 4 variants of the second note.

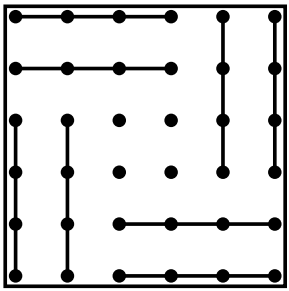
If the composer chooses the second and third notes for the accent tone, then there are 3 options for the first note.

Thus, in total, there are 8 different harmonic melody.

G. A circuit board

Time limit: 1 second
Memory limit: 256 MiB

The engineers are building a circuit board for a new supercomputer. The board has a rectangular grid of contacts with N rows and M columns. The distance between adjacent contacts is 1 mm. The engineers have a lot of wiring pieces of length L mm. Each of them can connect two contacts that are L mm apart. No two wires may touch, even at endpoints. Find the maximum possible number of wires that can be used.



Input
The numbers N, M, L, each a positive integer not greater than 1000.

Output
Output one integer, the answer.

Note
This problem has three subtasks, worth 20, 30, and 50 points, respectively. Points for each subtask are awarded only if the solution passes all the tests of that subtask.

Example

standard input	standard output
6 6 3	8

Note
The wires might not be parallel to the sides of the board.

H. Earthquake

Time limit: 2 seconds
Memory limit: 256 MiB

Another earthquake hit the city of Tubeville, and the residential water supply system stopped working.

Mario, the chief plumber of the city, still a youth then, designed Tubeville's water supply system to be earthquake-resistant. To achieve it, he broke the city area into square blocks and made the pipe sections in each block easily detachable from each other and rotatable around the center of the block into one of four positions.

Sadly, this architecture has the following problem: when an earthquake occurs, all pipe sections detach from each other and randomly turn around their block's center. To fix it, we need to return the pipe sections into position, preventing water from spilling, supplying all houses with water while preventing leaks. The design of the system precludes moving the pipes from one block to another.

Input

The first input line contains positive integers N and M describing the size of the Tubeville ($1 \leq N, M \leq 10$).

Each of the following N lines contains M numbers. Each number is a description of a city block $[i, j]$ ($0 \leq i < N$, $0 \leq j < M$).

A house that needs water supply is denoted by the number 2222, and the area with a water source is denoted by the number 3333. Otherwise, the number consists of four ones or zeroes, each representing the presence or absence of a connection to the neighbouring block, starting from the upper neighbour on the map in clockwise order.

Output

Your program must output N lines of M numbers matching the input data format. If there are several solutions, then output any of them. We guarantee the existence of the solution.

Note

The points for each test are awarded independently.

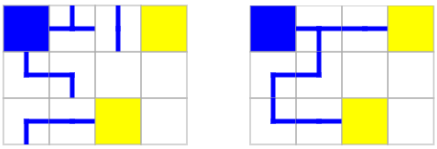
Example

standard input	standard output
3 4 3333 1101 1010 2222 1100 0011 0000 0000 0110 0101 2222 0000	3333 0111 0101 2222 0110 1001 0000 0000 1100 0101 2222 0000

Note

Some pipe segments may be spare and not used in the water supply system. In that case, they must not be connected to a water source to prevent water leaks. Houses cannot occupy an block adjacent to a water source. Also, houses cannot pass water through to other houses.

Scheme of the water supply system from the example:



I. The maze of the RoboBear

Time limit: 2 seconds
Memory limit: 256 MiB

Today Darina has received an email. It claims that the latest breakthrough of *Sakha Dynamics*, the RoboBear, has learned to pass mazes with the minimum number of turns along the way.

"Anyone who can go through this maze in fewer turns will receive a barrel of delicious coffee", the email says. Darina doubts if she needs so much coffee, but she asks you to help her calculate the minimum number of turns to pass the maze. Then, knowing the answer, she can decide whether she should try to beat the RoboBear.

Input

The first line contains two integers, N and M ($1 \le N, M \le 10^5, 1 \le N*M \le 10^5$).

Each of the following N lines contains M characters. Symbols `.` and `*` denote an empty square or a wall, respectively. The letter `E` denotes the exit

from the maze. The starting point is marked by one of the letters: U, D, L, R, depending on the direction that the robot is facing initially (Up, Down, Left, or Right, respectively).

Output
Your program must write positive integer number, the problem's answer, or −1 if RoboBear can not reach the exit without breaking the walls.

Note
The points for each test are awarded independently.

Examples

standard input	standard output
4 4 D.*. .**. ...* **.E	3
2 2 U. E.	2

J. 1D Arkanoid

Time limit: 4 seconds
Memory limit: 256 MiB

Sargylana is learning game development, her first project is Arkanoid in 1D. The game looks as follows: there is an infinite tape divided into square cells, some cells contain obstacles. You can put the ball into any cell without an obstacle and push it to the left or to the right. To simplify the problem, we consider the obstacles to be infinitely thin and the ball to be infinitely small. Every second the following happens:

- The ball moves to the next cell according to its current direction,
- If the cell contains an obstacle, then the ball changes its direction to the opposite, the next second it will begin moving in the reverse direction.

The obstacle it hit is destroyed, and the ball will be able to pass through it in the future.

To make sure there are no bugs in the game, Sargylana is going to the part of the game engine responsible for the ball's movement. She made some tests: given initial obstacle locations and many queries "if we put the ball in the cell x_i and push it left or right, where would it be after t_i seconds?". Note that all queries and independent: obstacles destroyed during a query will be present again for the next one.

Input

The first line contains two integers n and m ($1 \le n \le 10^6, 1 \le m \le 3 \cdot 10^5$) — the number of obstacles and the number of queries. The next line contains n integers $x_1 < x_2 < \ldots < x_n$ ($|x_i| \le 10^9$) — numbers of cells containing obstacles, in the increasing order. Next m lines contain queries. Every line contains a character L or R and two integers p_i and t_i ($|p_i| \le 10^9; 0 \le t_i \le 10^{18}$). If the character is L, then the ball starts moving to the left, if it's R, then the ball starts moving to the right.

It's guaranteed that the cell p_i does not contain an obstacle.

Output

For each query print an integer — the position of the ball in the i-th query after t_i seconds.

Note

This problem has five subtasks. Points for each subtask are awarded only if the solution passes all the tests of that subtask and all previous subtasks.

Subtask 1 (points: 16)
$n = 1, m = 1, t_i, |x_i|, |p_i| \le 100$.

Subtask 2 (points: 18)
$n, m \le 100, t_i, |x_i|, |p_i| \le 100$.

Subtask 3 (points: 24)
$n, m \le 1000, t_i, |x_i|, |p_i| \le 10^8$.

Subtask 4 (points: 29)
$n, m \le 50\,000$.

Subtask 5 (points: 13)
No additional constraints.

Examples

standard input	standard output
1 1 6 L 8 10	14
4 5 1 4 6 9 L 5 10 R 5 20 R 3 15 R 8 12 L -5 5	3 5 0 14 -10

Solutions

A. Necklace

Several different methods may solve this problem with varying degrees of efficiency.

It is possible to represent a necklace as a binary number of $2n$ digits, where ones and zeros correspond to the beads of different colors. Unfortunately, solutions of this kind are usually ineffective. They can only pass the tests of the first subtask for 20 points.

Another group of solutions uses the method of mapping. These solutions introduce an additional array to store the difference between blue and red beads. Such solutions work more efficiently. They can pass tests of two subtasks and get 40 points.

The most effective are solutions that apply formulas for Catalan numbers. As a result, they will successfully pass all tests and achieve the maximum score.

Let's explain the solution using an explicit formula for the Catalan numbers. Denote by $C(k)$ the number of variants for constructing a necklace with k beads. Consider first the case of an even number of beads k. Thus, $k-1$ is odd and, by the conditions of the problem, the kth bead can be of any color. That is, in the case of even k, we have

$$C(k) = 2C(k-1).$$

Now consider the case of odd k. Consequently, $k-1$ is even, and we can add a blue bead to the existing $k-1$ beads without breaking the rules.

However, if we add a red bead to the same $k-1$ beads, some variants will be excessive. These variants appear when we add a red bead to the equal number of blue and red beads.

Let $m = (k-1)/2$. It is known that if the number of red and blue beads is equal, and at the same time, the number of blue beads is always greater than or equal to red, then the number of variants for constructing a necklace is equal to the Catalan number $\frac{1}{m+1}\binom{m}{k-1}$ (where $\binom{a}{b}$ stands for binomial coefficient). Therefore, if k is odd:

$$C(k) = 2C(k-1) - \frac{1}{m+1}\binom{m}{k-1}.$$

Naturally, $C(1) = 1$ and $C(2) = 2$. Now, using the above recurrent formulas, you can easily calculate the number of necklace variants.

B. Orchestra

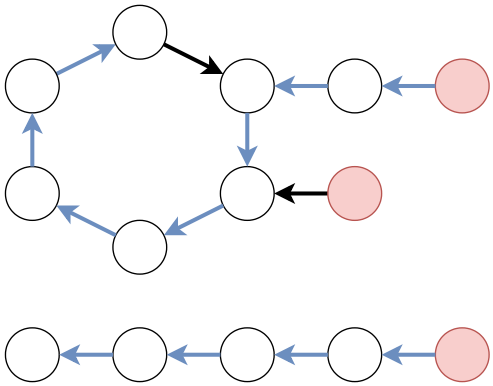
Let's build a directed graph where the musicians will correspond to the vertices.

After analyzing the conditions of the problem, it is easy to understand that there can't be more than one cycle in each connected component (for strict proof, try to construct a counterexample).

You can use a standard graph traversal algorithm (either breadth-first or depth-first search) to count the number of connectivity components. Notice, it is better to implement graph traversal in an iterative style since it will prevent stack overflow.

It is also easy to see that the graph often contains specific vertices that have no incoming arcs. Let's call them "tails" for definiteness. You can place such vertices into a separate list during the reading of input data.

Every chain is starting at one of such tails. To find the chain of maximal length, you can write a loop to check them. Inside it, you will need a WHILE-type loop to count the vertices of the chain.



C. Find and prevent

Let's select vertex *root* as the root of the tree (you can always choose vertex 1 as the root).

We need to count sums in all subtrees. That can be done with simple depth-first search algorithm. Denote the sum in a subtree with root i as $STS[i]$.

Now we can easily calculate required value for any vertex — if we remove edges connected to vertex i, transport connectivity can be found by the formula:

$$\frac{(STS[root] - STS[i])(STS[root] - STS[i] - 1)}{2} + \sum_{j \in \text{children[i]}} \frac{STS[j](STS[j] - 1)}{2}.$$

Final algorithm complexity is $O(N)$.

D. Expected xor

The main ideas of the full solution are:

- Mean xor can be computed by finding the probabilities for each of the 63 bits that they will be set to 1 and adding these probabilities weighted by 2^k, where k is the bit index.
- The probabilities can be computed independently of each other.
- Once we fix the bit k (where k is from 0 up to 62) whose probability that it will be set to 1 in the xor sum we want to compute, we can loop through the array and update this probability as we encounter new numbers.
- If the probability that the xor sum of the numbers in the array prefix considered so far contains bit k set to 1 is q, and the probability that the newly encountered number (by which we extend the aforementioned prefix) has 1 in bit k is p, the probability q' that the xor sum for the extended prefix is $p(1 - q) + (1 - p)q$.
- Initially, when the considered prefix is null, q is 0. Finally, when the considered prefix is the whole array, q is the required probability. As per a previous observation, we just add $q \times 2^k$ to the answer.
- To compute the probability p that a certain number b that is randomly chosen from the set $\{0, 1, \ldots, A - 1\}$ has bit k set to 1 is a matter of $O(1)$ calculation. Hints: $(A >> (k + 1) << k)$; if $(A\&(1 << k))$ consider $(A\&((1 << k) - 1))$.

- To work with modular probabilities, here it's enough to think about division by A as multiplication by *modular_inverse*(A). Since As are all coprime with the modulo number M, these modular inverses exist for each A and can be computed in $O(\log M)$ time after precomputing the number of numbers less than M that are coprime with M.

E. Odd Taxi

For each node, we want to determine how many participants will sit in a two-passenger cab with some of the descendants of the root of the subtree. To do this, let us calculate $dp[v][cnt]$ — the cost of the subtree v if we add cnt participants (participants without pairs from ancestors) to the root of the subtree. Note that in the optimal answer $cnt \leq$ number of leaves in subtree, otherwise there's a pair of participants in the subtree that lie on the same path from the root, it's more profitable to put them in one cab. How do we compute this value?

We need to match cnt participants with the descendants, this is a backpack problem, where $dp[u][cnt_1]$ is an item with weight cnt_1. This will add up to $\mathcal{O}(n^2)$. It remains to take into account the participants who were left without a pair at the root of the subtree. Suppose there are cnt left, you have to pay

$$depth[v] * \lfloor \frac{cnt}{2} \rfloor * y + depth[v] * (cnt \,\%\, 2) * x$$

(integer division and modulo remainder). These values with the same parity form an arithmetic progression, so you can calculate the values $dp[v][cnt]$ using $dp[v][cnt_1 < cnt]$ by maintaining two minima. The answer will be in $dp[1][0]$.

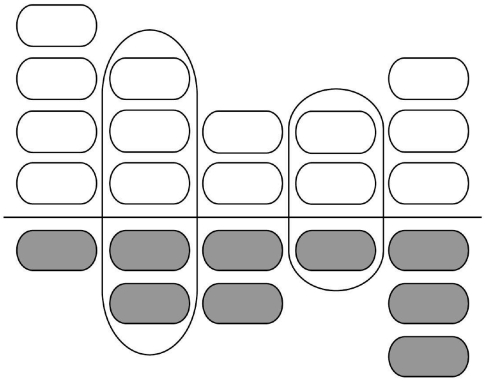
F. Alien music

After analyzing the condition of the problem, you can understand that the number of options can be large, so you should use the int64 (long long int) type variable to store it.

When reading the input data, you need to create two linear arrays: let denote them a and b. In the array a, we will store the number of notes with an accent tone. And in the array b — the number of other notes.

First, we create a double loop to check all the two-note variations with accent tone.

Then, for each selection of two notes with an accent tone, you need to count the number of options for all the remaining notes with other tones.



On this diagram two notes are selected ($i = 2, j = 4$). In this case, The number of options can be obtained by the formula:

$$b_1 \cdot a_2 \cdot b_3 \cdot a_4 \cdot b_5 = 4 \cdot 2 \cdot 2 \cdot 1 \cdot 3 = 48.$$

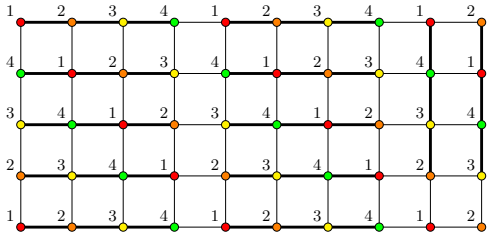
Here is a listing for the main part of the program on C:

```
long long int answer = 0;
for (int i=0; i<n; i++) {
   for (int j=i+1; j<n; j++) {

      long long int count = 1;
      for (int k=0; k<n; k++) {

         if ((k != i) && (k != j)) {
             count *= b[k];

         }
      }

      answer += a[i] * a[j] * count;
   }
}
```

G. A circuit board

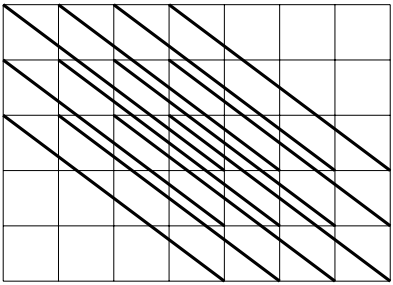
Check whether L can be the hypotenuse of a right-angled triangle with integral legs.

In case it cannot, the wires may only be parallel to the sides of the board. A greedy solution obtained for one of the two possible orientations will be optimal. We fill the available space with horizontal wires arranged in columns, going from left to right while possible. If space remains after that, we fill it with vertical wires arranged in rows similarly. This should be tried for both possible orientations of the board. An example for $L = 3$ on a board of size 5×10 is shown.



To prove the maximality of this solution, color the nodes along the diagonals into $L + 1$ colors. Because the wires may only be vertical or horizontal, each wire takes up exactly $L + 1$ nodes of different colors. So the greedy algorithm leaves the nodes of the colors that have more nodes. In the picture above, they are colors 1 and 2.

If L is a hypotenuse in an integral-side right-angled triangle, we must additionally consider the "diagonal" direction of the wires. If the legs are coprime, each wire takes up only 2 nodes. Horizontal or vertical wires will take up more nodes. So if the wires run diaganally parallel to each other, their number is only a little less than half the number of nodes. An example for $L = 5$ on a board of size 6×8 is shown.



Nonparallel wires will “get in each other’s way”. Thus, it is almost obvious that we should try running the wires diagonally parallel to each other in each of the possible directions.

H. Earthquake

To solve the problem you should organize an iterate over all available options. To optimize performance, you can take advantage of some features:

- the search does not include squares with values 0000, 1111, 2222, 3333;
- straight pipes can be aligned in the process of recursively checking the correctness of the pipeline;
- pipes cannot lead to 0000 and outside the city.

Thus, the enumeration involves cells with two and three ones, as well as pipes adjacent to the water source.

To check the correctness of the option, you should use a recursive bypass of the pipeline with the number of connected houses.

I. The maze of the RoboBear

To solve the problem, we need two basic things: dynamic programming and BFS.

The main thing of the problem is to calculate the minimum number of turns for each point individually, and then, knowing first points we reach from them other points.

Firstly, let us define a mechanism for visiting cells. When we arrive some cell, we memorize two parameters for that cell: how many turns we needed to get this and whether we arrived it — vertically or horizontally. If we visited that square both vertically and horizontally, we don't need to work on it later, because all the squares on the left, right, top, and bottom sides have already been visited.

Then, mark the movement from each cell to the next. If we have visited a square horizontally, we goes from here vertically, otherwise horizontally. For each cell visited from point A we put the number of turns to A plus 1.

The last one point: squares visiting order. The next square to visit is the square that needs the minimum number of turns we have available. By going around this way, we ensure that we take the minimum number of turns to get to each square we visit, because otherwise we would have another starting square that we haven't yet reached, and that has a smaller path than the current one, which can't be the case.

So, having understood these main points, we can safely solve this problem.

Problems asymptotic is $O(NLog(N))$ when prioritized via ordered_queue, and $O(N)$ when implemented carefully.

J. 1D Arkanoid

We're going to answer each query independently. First of all, let's figure out how many obstacles the ball is going to destroy. We will use binary search: to use it, we need to be able to find out how many seconds the ball will need to destroy k obstacles.

Without loss of generality, we will consider a ball that is pushed to the right. That ball will destroy $\lceil \frac{k}{2} \rceil$ obstacles to the right of its initial position, and $\lfloor \frac{k}{2} \rfloor$ obstacles to the left. Time required to do that can be calculated with a straightforward simulation (this solution gets 58 or 87 points, depending on how quick your solution is).

To get 100 points, we can use prefix sums. Let the positions of left obstacles be $x_1, \ldots, x_m$, and the positions of right obstacles are $y_1, \ldots, y_m$. The sum of times when the ball moved from left to right is $y_1 - p_i + y_2 - x_1 + \ldots + y_m - x_{m-1}$. Rearranging this sum, we get $y_1 + y_2 + \ldots + y_m - (p_i + x_1 + \ldots + x_{m-1})$, which can be calculated with prefix sums. Similarly, we can calculate the time it takes to move the ball of right to left. Implement this part carefully to deal with all the

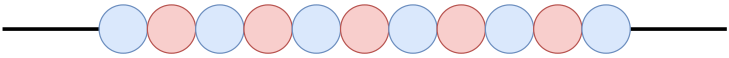
parity cases. Now, we can use binary search to calculate the time it takes to destroy these k obstacles, find the remaining time and find the final position of the ball. This solution works in $O(n + m \log n)$.

Задачи

A. Ожерелье

Ограничение по времени: 1 секунда
Ограничение по памяти: 1024 МБа

Ожерелье составлено из $2N$ красных и синих бусинок, при этом среди любых K бусинок, отсчитанных с его левого края, количество красных бусинок меньше или равно количеству синих.



Сколько различных ожерелий можно составить при этих условиях?

Формат входных данных

Целое положительное число N ($N \leqslant 32$).

Формат выходных данных

Целое положительное число — искомый ответ.

Система оценивания

Данная задача содержит три подзадачи. Баллы за каждый тест каждой подзадачи начисляются независимо.

Подзадача 1 (баллов: 20)
$N < 10$.

Подзадача 2 (баллов: 20)
$N < 20$.

Подзадача 3 (баллов: 60)
$N \leqslant 32$.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
1	2
2	6

B. Оркестр

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 МБ

Однажды в симфоническом оркестре заболел дирижер. Без него музыкантам очень трудно играть в одном ритме. Чтобы добиться этого, некоторые музыканты смотрят на одного из своих коллег и стараются подстроиться под него. Другие музыканты закрывают глаза и играют на слух.

Если проследить, кто на кого смотрит, то можно построить цепочки из музыкантов, которые последовательно смотрят друг на друга. Иногда такие цепочки зацикливаются. Будем считать, что музыкант не может повторно входить в одну и ту же цепочку.

К сожалению, наш оркестр пока мало репетировал и часто начинает играть не в такт, поскольку в нем образуются отдельные группы музыкантов, которые смотрят только друг на друга и не смотрят на музыкантов из других групп.

Чтобы научиться играть лучше, музыканты просят вас написать программу для анализа текущей ситуации.

Формат входных данных

В первой строке задано натуральное число N — количество музыкантов в оркестре ($N < 4 \cdot 10^4$).

В каждой из последующих N строк задано по одному номеру коллеги, на которого смотрит текущий музыкант. Если записано число 0, то музыкант ни на кого не смотрит.

Формат выходных данных

В первой строке выведите количество независимых друг от друга групп музыкантов K.

В следующих K строках выведите количество музыкантов в максимально длинной цепочке в каждой группе музыкантов. Эти числа нужно вывести в порядке возрастания.

Система оценивания

Баллы за каждый тест начисляются независимо.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
3 0 1 1	1 2
6 2 3 2 5 6 5	2 3 3

C. Найти и предотвратить

Ограничение по времени: 5 секунд
Ограничение по памяти: 256 МБ

Министерство Транспорта Древляндии узнало о готовящейся несанкционированной акции в одном из городов страны. Активисты из «Двусвязности» угрожают устроить транспортный коллапс, перекрыв въезды и выезды к одному из городов, чтобы обратить внимание на недостатки дорожной сети. Известно, что они хотят нанести максимальный вред транспортной доступности всей страны.

В Древляндии N городов, и, как вы могли догадаться, дорожная сеть представляет собой дерево. В i-м городе проживает a_i человек. Если есть несколько способов выбрать город, то активисты выберут тот, который имеет меньший номер.

Министерству недавно урезали финансирование, так что не нашлось ни одного специалиста, способного определить цель активистов, и поэтому они просят вас справиться с этой задачей.

Транспортная доступность всей страны равна количеству пар людей, которые могут добраться друг до друга, двигаясь по дорогам, или находятся в одном городе.

Формат входных данных

В первой строке задано одно целое число N ($1 \leqslant N \leqslant 1000000$).

Во второй строке заданы N целых чисел $a_1, a_2, \ldots, a_N$ ($1 \leqslant a_i \leqslant 100$).

Далее в $N - 1$ строках дано описание дорог. В i-й строке задано 2 целых числа — номера городов, между которыми есть дорога.

Формат выходных данных

В первой строке выведите город, в котором произойдет акция.

Во второй строке выведите значение транспортной доступности, если акцию не удастся предотвратить.

Система оценивания

Данная задача содержит три подзадачи. Баллы за подзадачу начисляются только, если все тесты этой подзадачи пройдены.

Подзадача 1 (баллов: 20)
$n \leqslant 1000$.

Подзадача 2 (баллов: 60)
$n \leqslant 100000$.

Подзадача 3 (баллов: 20)
$n \leqslant 1000000$.

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
3 1 1 1 1 2 2 3	2 0

D. Ожидаемый XOR

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 МБ

Согласно одной легенде, где-то в Румынии есть священное дерево, вокруг ствола которого записано число M. Под деревом сидит с огромной книгой старец и кидает N волшебных игральных костей. Каждая волшебная кость отличается от других и имеет A_i граней, пронумерованных

от 0 до $A_i - 1$, причем A_i взаимно просто с M (то есть наибольший общий делитель чисел A_i и M равен 1). После каждого броска костей старец смотрит на выпавшие на них числа b_i и записывает значение

$$b_1 \textbf{ xor } b_2 \textbf{ xor } \ldots \textbf{ xor } b_N$$

в книгу, где **xor** — операция побитового исключающего ИЛИ. Согласно легенде, комбинации на костях никогда не повторяются, и когда старец получит их все, наступит конец света. Вычислите среднее арифметическое чисел, которые будут записаны им в книгу к тому моменту, по модулю M, как описано ниже.

Формат входных данных
В первой строке даны целые положительные числа M, N ($N \leqslant 50000$, $2 \leqslant M \leqslant 10^9 + 7$). Во второй строке даны N положительных целых чисел A_i ($A_i \leqslant 2^{62}$, $i = 1, \ldots, N$).

Формат выходных данных
Выведите неотрицательное целое число X ($X < M$) такое, что если ответ — это дробь $\frac{U}{V}$ (где U и V — целые числа), то произведение X и V дает такой же остаток от деления на M, что и U:

$$X \cdot V \equiv U \pmod{M}.$$

Система оценивания
Данная задача содержит пять подзадач. Баллы за подзадачу начисляются только, если все тесты этой подзадачи пройдены.

Подзадача 1 (баллов: 20)
$N \leqslant 5, M = 11, A_i < 2^3$.

Подзадача 2 (баллов: 20)
$N \leqslant 100, M = 997, A_i \leqslant 2^5$.

Подзадача 3 (баллов: 20)
$N \leqslant 50000, M = 10^9 + 7, A_i \leqslant 2^{62}$.

Подзадача 4 (баллов: 20)
$N \leqslant 1000, M \leqslant 1000, A_i < 2^{30}$.

Подзадача 5 (баллов: 20)

$N \leqslant 50000, M \leqslant 10^9 + 7, A_i \leqslant 2^{62}$.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
11 1 10	10
10 3 7 9 3	8

Замечание

- В первом примере среднее арифметическое равно $\frac{9}{2}$, и выводится 10, так как $10 \cdot 2 = 20 \equiv 9 \pmod{11}$.
- Во втором примере среднее арифметическое равно $\frac{274}{63}$, и выводится 8, так как $8 \cdot 63 = 504 \equiv 4 \equiv 274 \pmod{10}$.

E. «Odd Taxi»

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 МБ

Олимпиада проходит в городе, дорожная сеть которого представляет собой подвешенное дерево из n пронумерованных вершин-домов. Место проведения олимпиады находится в вершине номер 1. Каждая из вершин с номером $i > 1$ связана дорогой с вершиной p_i ($p_i < i$).

Организатор олимпиады обязался развести всех участников по домам и для этого заключил договор с компанией под названием «Odd Taxi», которая предоставляет два вида услуг:

1. Перевозка одного пассажира, тариф — $dist_v * x$ рублей, где $dist_v$ — число дорог от вершины 1 до места назначения v.
2. Перевозка двух пассажиров, тариф — $dist_v * y$ рублей, причём таксист везет одного пассажира до вершины v и может высадить второго пассажира в любой вершине на пути до v, включая v.

Такси всегда едет кратчайшим путем; такси второго типа не может взять только одного пассажира.

Если в каждом i-м доме проживает a_i участников, какое минимальное количество денег нужно потратить организатору, чтобы все добрались до дома?

Формат входных данных

В первой строке даны число вершин n ($2 \leqslant n \leqslant 5000$), тарифы x и y ($1 \leqslant x \leqslant 10^5$, $1 \leqslant y \leqslant \min(2x - 1, 10^5)$) первого и второго типа такси, соответственно .

Во второй строке даны $n - 1$ чисел $a_2, \ldots, a_n$ — число участников в каждом доме ($0 \leqslant a_i \leqslant 10^5$).

В третьей строке даны $n - 1$ чисел $p_2, \ldots, p_n$ — родители каждой из вершин, начиная со второй ($1 \leqslant p_i < i$).

Формат выходных данных

Выведите единственное число — требуемое количество денег.

Система оценивания

Данная задача содержит три подзадачи. Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и предыдущих подзадач успешно пройдены.

Подзадача 1 (баллов: 20)
$n, a_i \leqslant 50$.

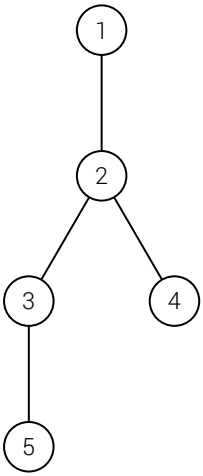
Подзадача 2 (баллов: 30)
$n \leqslant 500$.

Подзадача 3 (баллов: 50)
Нет дополнительных ограничений.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
5 2 3 2 0 1 1 1 2 3 3	15
5 3 2 1 0 1 1 1 2 2 3	12

Замечание



Дерево из второго примера. Выгоднее всего заказать такси второго типа для участников из вершин 2 и 5, первого типа для участника из вершины 4. В сумме заплатим $3 * y + 2 * x = 12$.

F. Инопланетная музыка

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 МБ

В будущем человечество нашло способ быстрого перемещения по галактике. Алиса — молодой исследователь из 22 века, которая занимается изучением инопланетной цивилизации Руматов.

В этой цивилизации существует аналог земной музыки. Алиса выяснила, что инопланетная музыка представляет собой мелодии из N последовательных нот. В отличие от земной музыки (в которой всего 7 нот), инопланетный композитор может выбрать из M_i вариантов ($1 \leqslant i \leqslant N$) для i-й ноты в мелодии.

Кроме того, для понимания инопланетной музыки важно знать, что каждая нота обладает тональностью, а мелодия в целом — акцентной тональностью. Для того, чтобы мелодия считалась гармоничной, требуется, чтобы в мелодии было ровно 2 ноты в акцентной тональности.

Напишите программу, которая при заданных ограничениях вычислит количество различных мелодий, соблюдающих правило гармонии инопланетной музыки.

Формат входных данных

В первой строке задано натуральное число $N \leqslant 15$ и номер акцентной тональности K ($0 < K \leqslant 20$).

Во второй строке заданы N натуральных чисел $M_i \leqslant 10$ — количество различных нот в мелодии.

В последующих N строках задано по M_i чисел, означающих тональность каждой ноты.

Формат выходных данных

Выведите количество возможных вариантов гармоничных мелодий.

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
3 1 4 5 2 1 2 3 4 1 2 3 3 4 1 2	8

Замечание

В мелодии будет 3 ноты. Акцент делается на тональность №1.

Для первой ноты есть 4 варианта, Для второй ноты — 5 разных вариантов (2 из них одной тональности) и 2 варианта третьей ноты.

Если композитор выбирает первую и второй ноты для акцентной тональности, тогда остается только один вариант третьей ноты.

Если композитор выбирает первую и третью ноты для акцентной тональности, тогда имеется 4 варианта второй ноты.

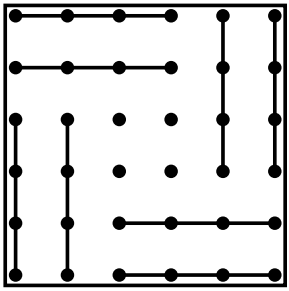
Если композитор выбирает вторую и третью ноты для акцентной тональности, тогда имеется 3 варианта первой ноты.

Таким образом, всего у композитора есть 8 различных вариантов мелодии.

G. Монтажная плата

Ограничение по времени: 1 секунда
Ограничение по памяти: 256 МБ

Инженеры размещают на большой прямоугольной монтажной плате элементы проектируемого суперкомпьютера. Плата состоит из отдельных контактов, образующих решётку из N горизонтальных и M вертикальных рядов. Расстояние по вертикали и по горизонтали между соседними контактами равно 1 мм. У разработчиков есть большое количество проводков длины L мм, каждый из которых можно припаять между контактами, расстояние между которыми тоже равно L мм. Любые два проводка не должны иметь общих точек, даже концов. Требуется рассчитать, какое наибольшее количество проводков можно разместить на плате.



Формат входных данных
N, M, L — натуральные числа, не превышающие 1000.

Формат выходных данных
Одно число — ответ.

Система оценивания
Данная задача содержит три подзадачи. Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи успешно пройдены.

Подзадача 1 (баллов: 20)

Подзадача 2 (баллов: 30)

Подзадача 3 (баллов: 50)

Пример входного и выходного файлов

стандартный ввод	стандартный вывод
6 6 3	8

Замечание

Проводки не обязательно должны размещаться параллельно краям платы.

H. Землетрясение

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 МБ

В городе Трубвилль произошло очередное землетрясение, и в городе нарушилось водоснабжение.

Главный сантехник города, Марио, еще в свои юные годы спроектировал трубопровод устойчивым к землетрясениям. Для этого он разбил город на квадратные участки, на которых и располагаются трубы. Соединяться могут только трубы смежных участков (смежными называются те участки, которые имеют общую сторону). Главная особенность водопровода заключается в том, что трубы легко отсоединяются друг от друга и вращаются относительно центра квадратного участка, на котором они расположены.

Но у такой архитектуры есть небольшой изъян: когда происходит землетрясение, все трубы отсоединяются друг от друга и случайным образом прокручиваются относительно центра. Чтобы починить водопровод, нужно соединить его фрагменты таким образом, чтобы вода поступала во все дома без протечек. Перемещать трубы на другой участок сантехники не могут.

Помогите сантехникам Трубвилля восстановить водоснабжение города.

Формат входных данных

В первой строке заданы целые числа N и M — размеры города Трубвилль ($1 \leqslant N, M \leqslant 10$). В последующих N строках записано по M чисел, где каждое число является описанием участка $[i, j]$ ($0 \leqslant i < N$ и $0 \leqslant j < M$). Если число равно 2222, то на этом участке расположен дом, в который необходимо подвести трубопровод. Участок, на котором расположен источник воды, обозначен числом 3333. В остальных случаях число состоит из четырех нулей или единиц, отвечающих за наличие соединения с четырьмя смежными на плане города участками, начиная с верхнего по часовой стрелке.

Формат выходных данных

Вывести в N строках по M чисел аналогично формату входных данных. Если решений несколько, то вывести любое из них. Гарантируется, что решение есть всегда.

Система оценивания

Баллы за каждый пройденный тест начисляются независимо.

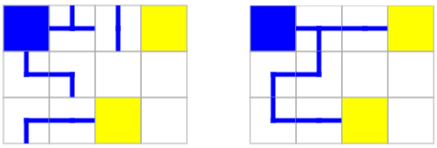
Пример входного и выходного файлов

стандартный ввод	стандартный вывод
3 4 3333 1101 1010 2222 1100 0011 0000 0000 0110 0101 2222 0000	3333 0111 0101 2222 0110 1001 0000 0000 1100 0101 2222 0000

Замечание

Некоторые фрагменты труб могут быть запасными и не использоваться в трубопроводе. Однако, они не должны быть соединены с источником воды, поскольку нельзя допускать протечку. Дома не могут находиться на соседнем участке с источником воды. Дома не могут проводить воду другому дому через себя.

Схематичное представление трубопровода из примера:



I. Лабиринт робомедведя

Ограничение по времени: 2 секунды
Ограничение по памяти: 256 МБ

Сегодня Дарине пришло письмо. В нем утверждается, что новейшая разработка компании «Саха дайнамикс» — робомедведь — научился проходить лабиринты за минимальное количество поворотов на пути.

Однако они хотят все-таки проверить алгоритм, заложенный в поведение робомедведя, и устраивают конкурс. «Любой, кто сможет пройти этот лабиринт за меньшее количество поворотов, получит бочку наивкуснейшего кофе», — говорится в письме.

Дарина обожает кофе, поэтому просит вас помочь ей — подскажите, за какое минимальное количество поворотов можно пройти весь лабиринт?

Формат входных данных

В первой строке записаны два целых числа N и M ($1 \leqslant N, M \leqslant 10^5$, $1 \leqslant NM \leqslant 10^5$). В следующих N строках задано по M символов. Каждый символ — это либо одна буква из набора `U`, `D`, `L`, `R` — точка старта с первоначальным направлением (вверх, вниз, налево, направо соответственно), либо `E` — выход из лабиринта, либо `*` — стенка, либо `.` — пустой проход.

Формат выходных данных

Выведите одно целое число, ответ, либо -1, если медведь не сможет выбраться из лабиринта, не ломая стен.

Система оценивания

Баллы за каждый тест начисляются независимо.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
4 4 D.*. .**. ...* **.E	3
2 2 U. E.	2

J. 1D Arkanoid

Ограничение по времени: 4 секунды
Ограничение по памяти: 256 МБ

Саргылана изучает разработку игр, и ее первым проектом стал одномерный арканоид. Игра в одномерный арканоид выглядит следующим образом: есть бесконечная полоска из квадратных клеток, в некоторых клетках есть препятствия. В любую свободную от препятствия клетку можно поместить шарик и толкнуть его влево или вправо. Для простоты будем считать, что препятствия имеют пренебрежительно малую толщину, а шарик — пренебрежительно малый диаметр. За одну секунду происходит следующее:

- Шарик за секунду переместится в следующую по направлению движения клетку.
- Если же в этой клетке оказалось препятствие, то шарик изменит свое направление на противоположное, и, начиная со следующей секунды, будет двигаться в обратном направлении. Препятствие при этом уничтожится, и шарик беспрепятственно пройдет через эту клетку в будущем.

Для того, чтобы в игре не было багов, Саргылана хочет протестировать часть игрового движка, моделирующую движение шарика. Для этого она создала несколько тестов: дано начальное расположение препят-

ствия, а также несколько запросов вида: «если шарик поместить в клетку x_i и толкнуть влево или вправо, где он окажется через t_i секунд?». Обратите внимание, что препятствия, уничтоженные во время исполнения запроса, будут восстановлены перед исполнением следующего.

Формат входных данных

В первой строке содержатся два числа n и m — число препятствий на полоске и число запросов ($1 \leqslant n \leqslant 10^6$, $1 \leqslant m \leqslant 3 \cdot 10^5$). В следующей строке содержатся n чисел $x_1 < x_2 < ... < x_n$ — номера клеток, в которых находятся препятствия, в возрастающем порядке ($|x_i| \leqslant 10^9$). Следующие m строк описывают запросы. Каждая строка содержит символ L или R и два числа p_i и t_i ($|p_i| \leqslant 10^9$; $0 \leqslant t_i \leqslant 10^{18}$). Если символ равен L, то шарик начинает свое движение влево, если R, то вправо.

Гарантируется, что в клетке p_i нет препятствия.

Формат выходных данных

Для каждого запроса выведите одно число — позицию шарика в i-м запросе через t_i секунд.

Система оценивания

Данная задача содержит пять подзадач. Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и предыдущих подзадач успешно пройдены.

Подзадача 1 (баллов: 16)
$n = 1, m = 1, t_i, |x_i|, |p_i| \leqslant 100$.

Подзадача 2 (баллов: 18)
$n, m \leqslant 100, t_i, |x_i|, |p_i| \leqslant 100$.

Подзадача 3 (баллов: 24)
$n, m \leqslant 1000, t_i, |x_i|, |p_i| \leqslant 10^8$.

Подзадача 4 (баллов: 29)
$n, m \leqslant 50\,000$.

Подзадача 5 (баллов: 13)
Нет дополнительных ограничений.

Примеры входного и выходного файлов

стандартный ввод	стандартный вывод
1 1 6 L 8 10	14
4 5 1 4 6 9 L 5 10 R 5 20 R 3 15 R 8 12 L -5 5	3 5 0 14 -10

Решения

A. Ожерелье

Задача может быть решена с разной степенью эффективности несколькими способами.

При первом способе используется перебор на базе двоичного числа длины $2n$. Здесь единицы и нули играют роль бусинок разного цвета. Решения такого типа, как правило, неэффективны, могут пройти лишь тесты первой подзадачи, которые оцениваются в 20 баллов.

При втором способе используется метод отображений. Здесь вводится дополнительный массив для подсчета разницы между синими и красными бусинками. Такие решения работают более эффективно, могут пройти тесты двух подзадач, и участники могут получить 40 баллов.

Наиболее эффективным способом является применение формул для чисел Каталана. Здесь решения участников проходят все тесты и оцениваются на максимальный балл.

Приведём разбор решения с применением явной формулы для чисел Каталана. Обозначим через $C(k)$ число вариантов построения ожерелья из k бусинок. Рассмотрим сначала случай чётного числа бусинок k. Тогда, $k-1$ — нечётно и, по условиям задачи, k-я бусинка может быть любого цвета. То есть, в случае чётного k, $C(k) = 2C(k-1)$.

Теперь рассмотрим случай нечётного k. То есть $k-1$ — чётно и к имеющимся $k-1$ бусинкам мы можем, не нарушая правил, добавить синюю бусинку.

Однако, если к этим же $k-1$ бусинкам добавляется красная бусинка, то некоторые варианты будут лишними. Они появляются при добавлении красной бусинки к бусинкам, где число синих и красных бусинок равно.

Пусть $m = (k-1)/2$. Известно, что если число красных и синих бусинок равно, и при этом, количество синих бусинок, всегда больше или равно чем красных, то количество вариантов построения ожерелья, равно числу Каталана $\frac{1}{m+1}\binom{m}{k-1}$ (где запись $\binom{a}{b}$ означает биномиальный коэффициент). Поэтому, в случае если k — нечётно:

$$C(k) = 2C(k-1) - \frac{1}{m+1}\binom{m}{k-1}.$$

Естественно, что $C(1) = 1$ и $C(2) = 2$. Теперь, используя приведённые рекуррентные формулы, можно легко вычислить количество вариантов ожерелья.

B. Оркестр

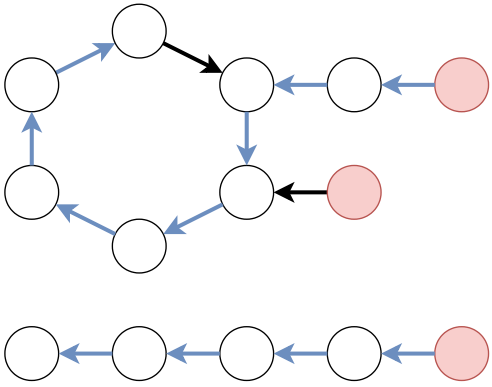
Построим ориентированный граф, в котором музыканты будут соответствовать вершинам.

После анализа условия задачи легко понять, что в каждой компоненте связности графа не может существовать больше одного цикла (для строгого доказательства попробуйте построить контрпример).

Для подсчета количества компонент связности можно использовать типовой алгоритм обхода графа (можно использовать как поиск в ширину, так и в глубину). Лучше реализовать обход графа в итеративном стиле, поскольку это поможет предотвратить переполнение стека.

Также легко видеть, что часто в графе будут присутствовать изолированные вершины, у которых нет входящих дуг. Назовем их для определённости «хвостами». Их можно поместить в отдельный список ещё на этапе чтения данных.

Каждая цепочка начинается с одного из таких хвостов. Чтобы найти цепочку максимальной длины, можно в цикле перебрать все хвосты, для каждого хвоста подсчитать количество вершин в цепочке.



C. Найти и предотвратить

Подвесим дерево за вершину *root* (за корень всегда можно принять вершину 1).

Посчитаем сумму по всем поддеревьям. Обозначим сумму в поддереве с корнем в i как $STS[i]$.

Теперь пройдемся по всем вершинам и найдем требуемое значение — при удалении ребер, соединенных с вершиной i транспортная доступность равна:

$$\frac{(STS[root] - STS[i])(STS[root] - STS[i] - 1)}{2} + \sum_{j \in \text{children[i]}} \frac{STS[j](STS[j] - 1)}{2}.$$

Итоговая асимптотика алгоритма — $O(N)$.

D. Ожидаемый XOR

Основные идеи полного решения вкратце таковы:

- Поскольку xor — операция поразрядная, то для нахождения среднего значения xor достаточно независимо друг от друга вычислить вероятности равенства единице для каждого из 63 двоичных разрядов, а затем сложить их с весами 2^k, где k — индекс разряда.
- Если рассматривать некоторую степень двойки k ($0 \leqslant k \leqslant 62$) и вычислить вероятность равенства единице k-го бита в сумме xor, можно двигаться по массиву A, обновляя значение вероятности по мере обработки последующих чисел.
- Если для данного k-го бита xor-суммы вероятность равенства единице для некоторого префикса массива равна q, а вероятность для нового числа A_j, которое мы присоединяем к префиксу, иметь единицу в k-м бите равна p, то вероятность q' для расширенного префикса равна $p(1 - q) + (1 - p)q$.
- Вначале, когда когда префикс пуст, $q = 0$. В конце, когда префикс равен всему массиву, q есть искомая вероятность для данного разряда. Как отмечено выше, к ответу мы просто добавляем $q \cdot 2^k$.

- Для вычисления вероятности того, что некоторое число b, наугад выбранное из множества $\{0, 1, \ldots, A-1\}$, имеет единицу в k-м бите, требуется вычисление сложности $O(1)$. Рассмотрим для примера

$$A = 87 = 10\mathbf{1}0111_2, \quad k = 4.$$

  Неотрицательных целых чисел, меньших A, с единицей в k-м бите существует $39 = 2 \cdot 16 + 7$. Первое слагаемое соответствует двум (10_2) группам вида $00\mathbf{1}****_2$ и $01\mathbf{1}****_2$, второе — числам от $10\mathbf{1}0000_2$ до $10\mathbf{1}0110_2$. Здесь жирным выделен интересующий нас k-й бит, а звездочками обозначены четыре разряда, которые могут быть заполнены любым из $16 = 2^k = 10000_2$ способов. В общем случае ответ наиболее компактно записывается с использованием побитовых операций И $\&$, сдвига влево $<<$ и вправо $>>$. Количество чисел в первой группе задается величиной $(A >> (k+1) << k)$, к которой следует прибавить $(A \,\&\, ((1 << k) - 1))$ в случае, если k-й бит числа A равен единице, т. е. если $(A \,\&\, (1 << k)) \neq 0$.

- Для нахождения ответа по модулю M следует заменить деление на A умножением на A^{-1}, где A^{-1} — это мультипликативный обратный к A элемент, то есть такое натуральное число, меньшее M, что произведение AA^{-1} дает остаток 1 по модулю M. Поскольку каждое A взаимно просто с модулем M, такое A^{-1} существует и может быть найдено за время $O(\log M)$, например, при помощи расширенного алгоритма Евклида, либо быстрого возведения A в степень $\varphi(M)-1$, где $\varphi(M)$ есть количество чисел, меньших M и взаимно простых с ним.

E. «Odd Taxi»

Для каждой вершины мы хотим как-то определить, сколько участников сядет в двухместное такси с кем-то из потомков корня поддерева. Для этого посчитаем $dp[v][cnt]$ — стоимость поддерева v, если добавить в корень поддерева cnt участников (участники без пар из предков). Заметим, что в оптимальном ответе $cnt \leqslant$ кол-во листьев в поддереве, иначе найдется пара вершин в поддереве, лежащих на одном пути от корня, их обоих выгодно посадить в двухместное такси. Как вычислить это значение?

Нам нужно набрать *cnt* участников из потомков. Это задача о ранце, где $dp[u][cnt_1]$ является предметом с весом cnt_1. Суммарно это сработает за $O(n^2)$. Осталось учесть участников, которые остались без пары в корне поддерева. Допустим их осталось *cnt*, за них надо заплатить

$$depth[v] * \lfloor \frac{cnt}{2} \rfloor * y + depth[v] * (cnt \,\%\, 2) * x$$

(целочисленное деление и взятие остатка по модулю). Эти стоимости образуют арифметическую прогрессию для одинаковых четностей, поэтому можно посчитать значения $dp[v][cnt]$ через $dp[v][cnt_1 < cnt]$, поддерживая два минимума. Ответ будет находиться в $dp[1][0]$.

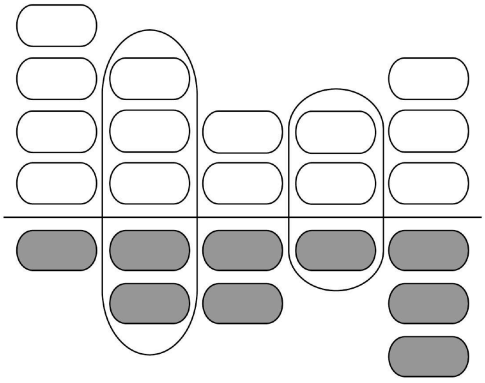
F. Инопланетная музыка

После анализа условия задачи можно понять, что количество вариантов может быть весьма велико, поэтому для их подсчета следует использовать тип int64 (long long int).

При чтении входных данных необходимо создать два линейных массива размера *N*: *a* и *b*. В массиве *a* будем хранить количество нот с акцентной тональностью, а в массиве *b* — количество остальных нот.

Сначала организуем двойной цикл, чтобы перебрать все варианты из двух нот акцентной тональности.

Затем, для каждого выбора двух нот с акцентной тональностью нужно подсчитать количество вариантов для всех оставшихся нот с другими тональностями.



На схеме выбраны две ноты (i = 2, j = 4). Количество вариантов в этом случае может быть получено по формуле:

$$b_1 \cdot a_2 \cdot b_3 \cdot a_4 \cdot b_5 = 4 \cdot 2 \cdot 2 \cdot 1 \cdot 3 = 48.$$

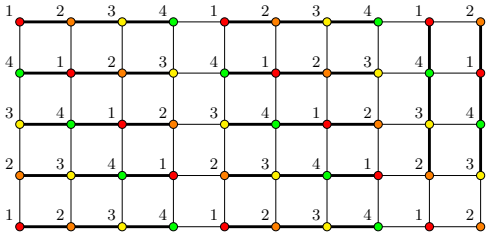
Приведем листинг для основной части программы на языке C:

```
long long int answer = 0;
for (int i=0; i<n; i++) {
   for (int j=i+1; j<n; j++) {

      long long int count = 1;
      for (int k=0; k<n; k++) {

         if ((k != i) && (k != j)) {
             count *= b[k];

         }
      }

      answer += a[i] * a[j] * count;
   }
}
```

G. Монтажная плата

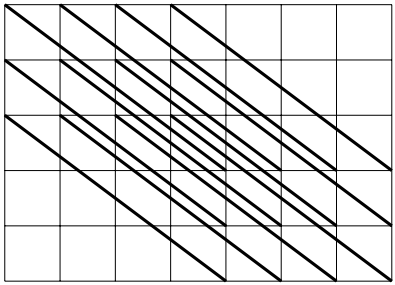
Выясняем, является ли L гипотенузой прямоугольного треугольника с целыми катетами.

Если не является, то расположение проводков должно быть строго параллельно краям платы. Оптимальным при этом является расположение, получаемое жадным алгоритмом. Он заключается в том, что сначала проводятся проводки подряд в верхней горизонтали слева направо, затем в следующей и т. д. После проведения горизонтальных проводков таким же образом проводятся и вертикальные. Такой процесс следует провести для каждой из двух возможных ориентаций платы. Пример при $L = 3$ на плате 5×10 приведён на рисунке.



Доказать оптимальность можно так. Покрасим узлы в $L + 1$ различных цветов по диагоналям. Так как проводки могут быть расположены только вертикально или горизонтально, то легко видеть, что каждый проводок занимает $L + 1$ узел разных цветов. Жадный алгоритм оставляет свободными узлы, которых больше. В случае на рисунке – это узлы 1-го и 2-го цветов.

Если L является гипотенузой прямоугольного треугольника с целыми катетами, то, кроме прямого расположения проводков, описанного в предыдущем случае, надо рассмотреть все возможные «диагональные» расположения проводков. При этом каждый проводок исключает из рассмотрения только два узла сетки. При прямом расположении каждый проводок занимает большее количество узлов. Таким образом, если проводки расположены параллельно друг другу, то их общее количество лишь немного меньше общего количества узлов, делённого на 2. Ниже приведен пример при $L = 5$ на плате 6×8.



Если же среди проводков есть непараллельные, то они «мешают» друг другу. Таким образом, почти очевидно, что в этом случае следует для каждого возможного направления располагать проводки по «диагоналям» вдоль этого направления.

H. Землетрясение

Для решения задачи необходимо организовать перебор вариантов. Для повышения производительности можно учесть следующее:

- в переборе не участвуют квадраты со значениями 0000, 1111, 2222, 3333;
- прямые трубы можно выравнивать в процессе рекурсивной проверки корректности трубопровода;
- трубы не могут уводить в 0000 и за пределы города.

Таким образом, в переборе участвуют клетки с двумя и тремя единицами, а также трубы, соседствующие с источником воды.

Для проверки корректности варианта следует использовать рекурсивный обход трубопровода с подсчетом подключенных домов.

I. Лабиринт робомедведя

Для решения задачи нам понадобится два основных метода: динамическое программирование и поиск в ширину.

Суть задачи в том, чтобы подсчитать минимальное количество поворотов для каждой точки по отдельности, а далее, зная точки, которые мы достигаем из каждой клетки, высчитывать минимальное количество поворотов для нее.

Для начала определим механизм посещения клеток. Когда мы приходим в определенную клетку, мы запоминаем для этой клетки два параметра: какое количество поворотов нам понадобилось для этой клетки и как мы к ней пришли — вертикально или горизонтально. Если мы посетили эту клетку и вертикально, и горизонтально, то в дальнейшем обрабатывать эту клетку не нужно, ведь все клетки слева, справа, сверху и снизу уже были посещены.

Далее обозначим движение от каждой клетки до следующей. Когда мы посетили клетку горизонтально, мы пускаем от нее посещения по вертикали, иначе — по горизонтали. Для каждой посещенной из точки A клетки запоминаем количество поворотов до A, увеличенное на 1.

И, наконец, определим очередь посещения клеток. Следующая на посещение клетка — это клетка, до которой нужно минимальное количество поворотов из всех доступных нам. Обходя путь таким образом, мы гарантируем, что до каждой посещенной нами точки мы потратили минимальное количество поворотов, ведь иначе у нас была бы другая, еще не обработанная стартовая клетка, путь до которой меньше текущей, чего не может быть.

Таким образом, поняв эти основные моменты, мы можем спокойно решить эту задачу.

Асимптотика задачи $O(N \log N)$ при задаче приоритета через структуру данных ordered_queue, при аккуратной реализации — $O(N)$.

J. 1D Arkanoid

Будем отвечать на каждый запрос независимо. Для начала узнаем, сколько препятствий разрушит шарик. Для этого будем использовать двоичный поиск: нужно уметь проверять, за сколько секунд шарик разрушит k препятствий.

Пусть, не умаляя общности, шарик изначально движется вправо. Мы знаем, что он разрушит $\lceil \frac{k}{2} \rceil$ препятствий справа от начальной позиции, и $\lfloor \frac{k}{2} \rfloor$ препятствий слева от начальной позиции. Время, нужное, чтобы пройти этот путь, можно вычислить моделированием (и получить 58 или 87 баллов в зависимости от аккуратности).

Для того, чтобы получить 100 баллов, нужно использовать префиксные суммы. Пусть координаты позиций препятствий слева это $x_1, \ldots, x_m$, а препятствий справа — $y_1, \ldots, y_m$. Сумма всех времен движения шарика слева направо равна $y_1 - p_i + y_2 - x_1 + \ldots + y_m - x_{m-1}$. Переписав эту формулу, получим $y_1 + y_2 + \ldots + y_m - (p_i + x_1 + \ldots + x_{m-1})$, что легко вычисляется префиксными суммами. Аналогично (не забыв про разные случаи в зависимости от четности k), можно вычислить время, которое шарик тратит на движение справа налево. Далее, мы умеем двоичным поиском находить число сломанных препятствий, находим оставшееся время, и вычисляем финальную позицию. Решение работает за $O(n + m \log n)$.

ИТОГИ / *FINAL STANDINGS*

№ #	Имя, фамилия *Name*	Страна *Country*	Команда *Team*	Задачи / *Tasks*										ИТОГО *TOTAL*	НАГРАДА *AWARD*
				1 день / *Day 1*					2 день / *Day 2*						
				A	B	C	D	E	F	G	H	I	J		
1	Ilie Daniel Apostol	Romania	"Tudor Vianu" National College of Computer Science	100	100	100	100	100	100	100	60	100	100	**960**	1DG, HML
2	Taimas Korganbayev	Kazakhstan	Almaty city	100	100	100	100	100	100	20	92	100	100	**912**	1DG, HML
3	Altair Ashurov	Kazakhstan	AOO "Nazarbayev Intellectual School"	100	100	100	100	50	100	20	84	100	100	**854**	1DG
4	Sng James (Sun Jie)	Singapore	NUS High School of Mathematics and Science	100	100	100	100	100	100	20	0	100	100	**820**	2DG, HML
5	Tudor Ivan	Romania	"Tudor Vianu" National College of Computer Science	100	100	100	100	0	100	100	16	100	100	**816**	2DG
6	Mukhammadarif Sakhmoldin	Kazakhstan	AOO "Nazarbayev Intellectual School"	100	100	100	100	20	100	20	72	100	100	**812**	2DG
7	Carol Luca Gasan	Romania	Saint Sava National College	100	100	100	100	20	100	20	60	100	100	**800**	2DG
8	Flaviu-Cristian Verde	Romania	"Tudor Vianu" National College of Computer Science	100	100	100	100	20	100	20	36	100	100	**776**	2DG
9	Stefan Popa	Romania	"Tudor Vianu" National College of Computer Science	100	100	100	100	0	100	20	40	100	87	**747**	2DG

10	Ramazan Perdebai	Kazakhstan	Karaganda region	100	100	100	100	0	100	20	68	100	58	**746**	2DG
11	Nikita Barykin	Russia	SSC NSU (Physics and Mathematics School at Novosibirsk State University)	100	100	100	100	.	100	20	56	100	58	**734**	2DG
12	Arsen Muzhikov	Kazakhstan	NJSC "Republican Physics and Mathematics School" Nur-Sultan city	100	100	100	100	.	100	20	0	100	100	**720**	2DG
13	Kanysh Mukhamedkarim	Kazakhstan	East Kazakhstan Region	100	100	100	80	0	100	20	0	100	100	**700**	2DG
14	Dinmukhamed Oralkhanov	Kazakhstan	East Kazakhstan Region	100	100	100	60	.	100	20	.	100	100	**680**	3DG
15	Egor Shevchenko	Russia	SSC KSU ("IT-lyceum" of Kazan Federal University)	100	100	100	60	0	100	20	.	100	100	**680**	3DG
16	Nurstan Duisengaliyev	Kazakhstan	Almaty city	100	100	100	60	0	100	20	0	100	100	**680**	3DG
17	Imran Turganov	Kazakhstan	NJSC "Republican Physics and Mathematics School", Almaty city	100	100	100	60	0	100	20	40	30	100	**650**	3DG
18	Joseph Oliver Lim	Indonesia	Wardaya College	100	100	100	20	.	100	0	28	100	100	**648**	3DG
19	Akram Rakhmetulla	Kazakhstan	NJSC "Republican Physics and Mathematics School" Nur-Sultan city	100	100	100	40	0	100	20	48	70	58	**636**	3DG
20	Timur Zykov	Russia	1st national team, Republic of Sakha (Yakutia)	100	100	100	.	0	100	100	0	100	.	**600**	3DG, SP

21	Yernar Sadybekov	Kazakhstan	AOO "Nazarbayev Intellectual School"	100	100	0	40	20	100	50	12	100	58	**580**	HMLP
22	Mikhail Ponomaryov	Kazakhstan	NJSC "Republican Physics and Mathematics School" Nur-Sultan city	100	100	100	.	0	100	20	0	100	58	**578**	HMLP
23	Margarita Petrova	Russia	Center for the preparation of the Olympic team at the Lyceum Boarding School No. 61	100	95	100	.	.	100	20	.	30	58	**503**	HMLP
24	Temirlan Zharylgamyssov	Kazakhstan	Nur-Sultan city	100	90	0	20	.	100	20	16	70	87	**503**	HMLP
25	Amir Nuriyev	Kazakhstan	AOO "Nazarbayev Intellectual School"	100	35	80	.	.	100	20	0	100	58	**493**	
26	Arman Issayev	Kazakhstan	Nur-Sultan city	100	90	0	.	.	100	20	.	80	58	**448**	
27	Erjan Takauov	Kazakhstan	Aktobe Region	55	95	0	.	.	100	20	8	80	87	**445**	
28	Vladislav Pavlovsky	Russia	2nd national team,Republic of Sakha (Yakutia)	100	.	0	.	.	100	20	.	100	58	**378**	SP
29	Vasily Parnikov	Russia	1st national team, Republic of Sakha (Yakutia)	100	25	0	.	.	100	20	16	40	58	**359**	SP
30	Beknur Les	Kazakhstan	Jambyl region	100	100	100	20	.	4	.	.	.	.	**324**	

31	Karim Suleimanov	Russia	SSC KSU ("IT-lyceum" of Kazan Federal University)	100	15	.	20	.	100	20	.	.	34	**289**	
32	Vlad Stefan Stancu	Romania	Liceul Teoretic Național	30	35	.	20	0	68	.	.	0	87	**240**	
33	Erik Paskalev	Bulgaria	National High School of Science and Mathematics "Akad. Lubomir Chakalov"	30	50	20	.	0	100	20	.	.	16	**236**	
34	Anai Kassymkhan	Kazakhstan	Pavlodar region	40	35	.	.	.	96	20	.	20	16	**227**	
35	Lena Uvarovskaya	Russia	1st national team, Republic of Sakha (Yakutia)	30	.	.	.	.	100	.	.	.	58	**188**	
36	Darkhan Semyonov	Russia	2nd national team, Republic of Sakha (Yakutia)	100	.	.	.	.	.	20	.	.	58	**178**	
37	Arkady Dmitriev	Russia	1st national team, Republic of Sakha (Yakutia)	100	.	.	.	.	.	.	.	.	58	**158**	
38	Bolashak Kulmuhambetov	Kazakhstan	Aktobe Region	20	20	.	.	.	68	20	.	30	0	**158**	
39	Georgy Gritchenko	Russia	St. Petersburg State University Academic Gymnasium named after D.K. Faddeev	0	.	.	.	.	100	20	.	.	34	**154**	
40	Andrey Kuzmin	Russia	1st national team, Republic of Sakha (Yakutia)	0	0	.	.	.	100	20	.	.	.	**120**	

41	Miron Makarov	Russia	2nd national team, Republic of Sakha (Yakutia)	0	.	.	.	.	.	20	.	.	58	**78**	
42	Ksenia Frolova	Russia	2nd national team,Republic of Sakha (Yakutia)	0	10	.	.	.	44	0	12	.	.	**66**	
43	Duolan Gavriliev	Russia	2nd national team, Republic of Sakha (Yakutia)	20	10	.	.	.	4	20	0	.	.	**54**	
44	Taisiya Okoneshnikova	Russia	2nd national team, Republic of Sakha (Yakutia)	0	5	.	.	.	.	0	.	.	.	**5**	
45	Ilyas Alimbekov	Kazakhstan	Jambyl region	0	.	.	.	.	.	.	.	.	.	**0**	
46	Muhammad Alfatih Arrazy	Indonesia	Wardaya College	0	0	0	0	0	.	0	.	.	.	**0**	

НАГРАДЫ / *AWARDS* :
1DG — диплом 1 степени / *1st degree diploma,*
2DG — диплом 2 степени / *2nd degree diploma,*
3DG — диплом 3 степени / *3rd degree diploma,*
HML — почетная грамота за полное решение задач 1 дня / *Honourable mention letter "For the perfect score of Day 1",*
HMLP — почетная грамота за многообещающие результаты / *Honourable mention letter "For the promising results",*
SP — спецприз Якутского отделения Дальневосточного центра математических исследований / *special prize of the Sakha branch of "Far Eastern Center for mathematical research"*

Contents | Содержание