



Drop-It! (dropit)

William has just downloaded the new *Drop-It!* game on his smartphone. In this game several blocks of different lengths are dropped by a crane one on top of the other. While falling, they follow these rules:

1. If a block falls on top of another one of equal length, the fall stops and both blocks involved are destroyed (they disappear);
2. If a block falls on top of a longer block, the fall stops and an extra block appears on top of the last one, with length equal to the length difference of the two blocks involved;
3. If a block falls on top of a shorter block B , its length is reduced by the length of B , B is destroyed and the fall continues.

Help William plan his strategy, by writing a program that computes the stack produced by the falling of N blocks of length L_i where $i = 0, \dots, N$.

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` or `.pas`.

 Among the attachments of this task you will find a template (`dropit.c`, `dropit.cpp`, `dropit.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following function:

C/C++	<code>int cadi(int N, int L[], int P[]);</code>
Pascal	<code>function cadi(N: longint; var L, P: array of longint): longint;</code>

Where:

- N is the number of blocks falling from the crane.
- L is an array indexed from 0 to $N - 1$, containing the lengths of the blocks dropped by the crane (in falling order).
- P is an array indexed from 0 to $M - 1$, that must be filled with the lengths of the blocks that form the final stack (in the same order as the input's).
- The function shall return M , the number of blocks in the final stack. This number will be printed to the output file along with the contents of the P array.

Input

File `input.txt` consists of two lines. Line 1 contains integer N . Line 2 contains integers L_i separated by spaces.

Output

File `output.txt` consists of two lines. Line 1 contains integer M . Line 2 contains integers P_i separated by spaces.



Constraints

- $1 \leq N \leq 100\,000$.
- $1 \leq L_i \leq 10\,000$ for all $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.

- **Subtask 1 [10 punti]:** Sample test cases.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N \leq 1000$.
- **Subtask 4 [30 punti]:** No limits.

Examples

input.txt	output.txt
4 42 42 42 20	3 42 20 22
4 17 13 4 15	3 17 2 15

Explanation

In the **first sample test case**, rule (1) is applied first, thus clearing the current stack. Then rule (2) is applied, thus producing an extra block of length 22.

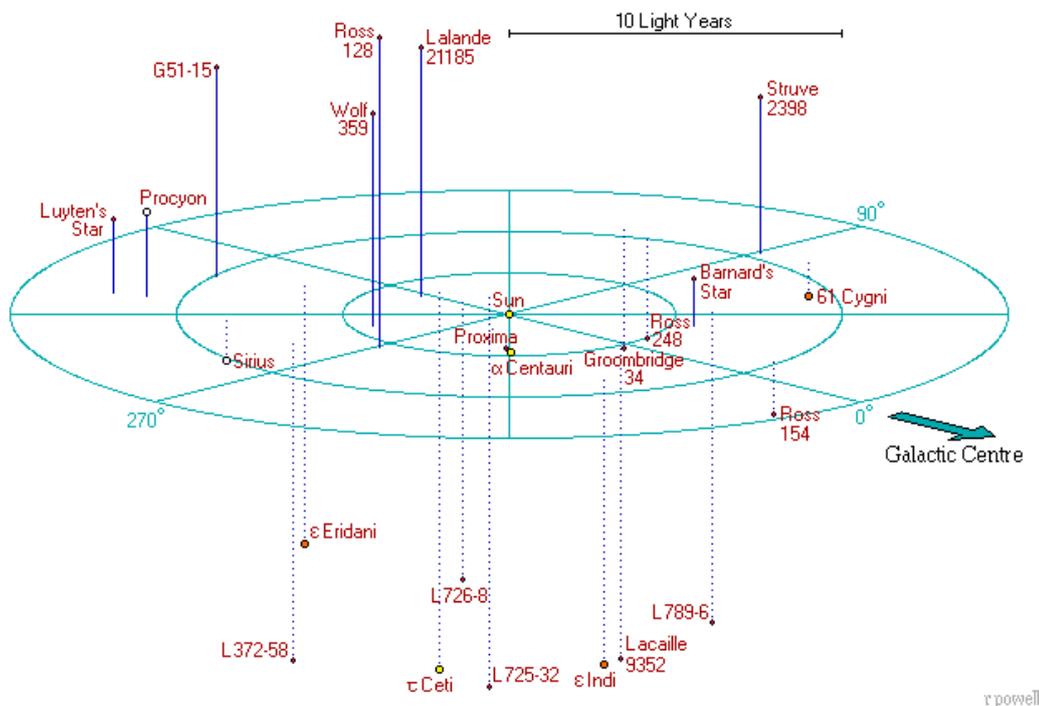
In the **second sample test case**, rule (2), (1), (3) and (2) get applied in order, thus obtaining the following succession of states:

```
17 <-- 13
17 13 4 <-- 4
17 13 <-- 15
17 <-- 2
17 2 15
```

Anno luce (annoluce)

A recent study confirmed the existence of gravitational waves. This extraordinary discovery has led more and more people to become interested in space. Sadly, space is still quite unreachable for common people. William is a bit depressed by this fact, but he's positive that he can exploit this media attention to advertise a new business. He decided to launch a startup for interstellar travel.

It's important to note, though, that stars can be quite far. The *Sun* is the star that's closest to us, but the others are pretty far. *Proxima Centauri*, the "second nearest star", is 4.24 light years away from the Sun: that means that it would take well over four years to reach this star! (assuming we know how to travel at the speed of light).



There are 33 stars within 12.5 light years from the Sun
 (Richard Powell CC BY-SA 2.5 via Wikimedia Commons)

William believes he can build a spaceship that is able to travel at the speed of light (he's found a YouTube tutorial that seemed quite trustworthy), so he went on and bought a telescope to trace a 3D map of the Milky Way. Each star is specified in the 3D map through a (x, y, z) point in space. The Sun is always in the map, and it's always specified by $(0, 0, 0)$.

Write a program that, given the 3D sky map, is able to answer Q queries: each query has an integer D , and must be answered with the number of stars that can be reached assuming that D will be spent for travelling.

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` or `.pas`.

📎 Among the attachments of this task you will find a template (`annoluce.c`, `annoluce.cpp`, `annoluce.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following functions:

C/C++	<pre>void mappatura(int N, int X[], int Y[], int Z[]); int query(int D);</pre>
Pascal	<pre>procedure mappatura(N: longint; var X, Y, Z: array of longint); function query(D: longint) : longint;</pre>

Where:

- The integer N is the number of stars in the sky map.
- Arrays X , Y and Z , indexed from 0 to $N - 1$, contain the coordinates of the N stars. That is: the i -th star is located at $(X[i], Y[i], Z[i])$.
- The `mappatura` function will only be called once, at the start of the program.
- Each call to `query(D)` must return the number of stars within D light years from the Sun.

Input

File `input.txt` consists of $N + Q + 2$ lines. The first line contains the single integer N . The next N lines contain the coordinates X_i, Y_i, Z_i of the i -th star, separated by spaces. The next line contains the single integer Q . The next Q lines contain the values of D for the relevant queries.

Output

File `output.txt` consists of Q lines containing an integer each: the answer for the relevant query.

Constraints

- $1 \leq N, Q \leq 100\,000$.
- $0 \leq X_i, Y_i, Z_i < 2^{30}$ for each $i = 0 \dots N - 1$.
- The unit of measure for x, y, z axes is the light year.
- $0 \leq D < 2^{31}$ for each call to `query(D)`.
- The D value is expressed in light years.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.



- **Subtask 1 [10 points]:** Sample test cases.
- **Subtask 2 [20 points]:** $Y[i] = Z[i] = 0$ for each i . Instead of being in a tridimensional space, every star lie on the same line!
- **Subtask 3 [20 points]:** $Z[i] = 0$ for each i . Instead of being in a tridimensional space, we're in bidimensional plane!
- **Subtask 4 [10 points]:** $N, Q, D \leq 10$.
- **Subtask 5 [10 points]:** $N \leq 100; D < 10\,000$.
- **Subtask 6 [10 points]:** $Q \leq 100; D < 10\,000$.
- **Subtask 7 [20 points]:** No limits.

Examples

input.txt	output.txt
3 0 0 0 1 1 1 2 2 2 5 0 1 2 3 4	1 1 2 2 3
5 1 2 3 4 5 6 0 0 0 9 9 9 7 1 1 3 20 8 9	5 3 4

Explanation

In the **first sample test case**, the 3 stars are respectively: 0 light years, $\sqrt{3}$ light years and $2\sqrt{3}$ light years away from the Sun.

In the **second sample test case**, the 5 stars are respectively: $\sqrt{14}$ light years, $\sqrt{77}$ light years, 0 light years, $9\sqrt{3}$ light years and $\sqrt{51}$ light years away from the Sun.

Enciclopedia olimpica (enciclopedia)

Giorgio has always been a fan of encyclopedias. Among his vast collection, some of the rarest encyclopedias in existence can be found. The most famous encyclopedia to ever be printed is probably the eleventh edition of the Encyclopaedia Britannica. Obviously, Giorgio owns all 29 volumes of it:



Encyclopaedia Britannica - eleventh edition
 (CC BY-SA 3.0 via Wikimedia Commons)

Giorgio has decided to create his own new encyclopedia. Instead of words, this “Olympic Encyclopaedia” will use the *names of the tasks* given during the previous editions of the Italian Olympiads in Informatics. The “definition” of each term will actually be a detailed tutorial for solving the corresponding task. For example, in the entry for “poldo”, Giorgio will write a detailed tutorial for solving the “La dieta di Poldo” task from the *regional selection 2004* of the Olympiads.

The spine of each volume contains an *indication* of which words can be found on the inside. For the purposes of this task, for a given volume that starts with a word p and ends with a word q , we’ll define such indication as a pair (p^*, q^*) of *minimal length* that satisfies the following criteria. First of all, p^* must be a prefix of word p that *is not* a prefix of q , and q^* must be a prefix of word q that *is not* a prefix of p . If one or both conditions is impossible (e.g. with $p = \text{“ciao”}$ and $q = \text{“ciaone”}$, we have $q^* = \text{“ciao”}$ but p^* doesn’t exist) then we’ll say that those prefixes are set equal to the entire word (in the given example: $p^* = p$).

For example, if the first and last word of the i -th volume are respectively *caldaia* and *carrucola*, then the spine of the i -th volume will need to say CAL-CAR. In this way, if we’re looking for the *cantina* task, we know that it will be inside the i -th volume.

However, the pair will need to satisfy another additional criteria: the p^* prefix must also *not* be a prefix of the previous volume's last q word (if there's such volume), and the q^* prefix must also *not* be a prefix of next volume's first p word (if there's such volume).

For example, if the $(i + 1)$ -th volume began with the word **cartella**, then we would need to replace the **CAL-CAR** indication of the i -th volume with **CAL-CARR** in order to avoid ambiguity. In the same way, if the $(i - 1)$ -th volume ended with the word **calamaro**, then we would need to change the **CAL-CARR** indication to **CALD-CARR**.

Giorgio has a list of N task names, one for each task in any past edition of the Olympiads (it's guaranteed that all task names are different), and he has already decided that the encyclopedia will be formed by K volumes (it's guaranteed that K divides N). Help Giorgio decide the indications to print on each volume's spine, so that he can order the K covers from his favorite workshop.

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` o `.pas`.

📎 Among the attachments of this task you will find a template (`enciclopedia.c`, `enciclopedia.cpp`, `enciclopedia.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following functions:

C/C++	<pre>void rilega(int N, int K, char* parole[]); void volume(char* S);</pre>
Pascal	<pre>procedure rilega(N, K: longint; var parole: array of ansistring); procedure volume(var S: ansistring);</pre>

Where:

- The integer N is the number of task names that will end up in the encyclopedia.
- The integer K is the number of volumes that will form the encyclopedia.
- The `parole` array, indexed from 0 to $N - 1$, contains the task names.
- The function does not return any value, instead, it should call K times the helper function `volume` to print the indication for a volume.
- The S parameter of the `volume` function is the indication that will be printed on the spine of the volume. It should be formed by two groups of lowercase alphabetic letters, and the groups should be separated by a “-” character. These groups of letters should be “lexicographically increasing”, that is, the first group should be lexicographically smaller than the second one.

Input

File `input.txt` consists of $N + 1$ lines. The first line has two integers N and K separated by a space. The next N lines have the words that Giorgio wants to put in the encyclopedia, one for each line.

Output

File `output.txt` consists of K lines containing the indications that must be printed on the spine of the Olympic Encyclopaedia's volumes.

Constraints

- $4 \leq N \leq 10\,000$.
- K strictly divides N (hence $1 \leq K < N$).
- The N names are formed by lowercase alphabetic characters only: no spaces.
- The N names will be provided in a lexicographically sorted order and without repetitions.
- $1 \leq |\text{parola}[i]| \leq 30$ for each $i = 0 \dots N - 1$. That is: the length of each word is ≤ 30 .

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.

- **Subtask 1 [10 punti]:** Sample test cases.
- **Subtask 2 [40 punti]:** $K = 2$.
- **Subtask 3 [20 punti]:** $N \leq 10$.
- **Subtask 4 [30 punti]:** No limits.

Examples

input.txt	output.txt
<pre> 9 3 anello barca calamaro caldaia cantina carrucola cartella dinosauri energia </pre>	<pre> A-CALA CALD-CARR CART-E </pre>
<pre> 4 2 xyzxx xzqh yay zzzzzzzzz </pre>	<pre> XY-XZ Y-Z </pre>
<pre> 4 2 a ab abc abcd </pre>	<pre> A-AB ABC-ABCD </pre>

Codici interessanti (interessante)

Giorgio is now studying the *interesting codes*, that is, sequences of N digits 0 or 1 such that for any possible ratio $x = 1, \dots, 10$ and starting value $i = 0, \dots, N - 3x - 1$, the digits in the positions given by the corresponding arithmetic progression $(i, i + x, i + 2x, i + 3x)$ are *not all equal to each other*. For example, the following 10-digit codes are interesting:

```
0001001000
0001100011
1010110111
```

While these are not:

```
1011101011
1000010110
1101000100
```

because of the arithmetic progressions (i, x) respectively $(0, 3)$, $(1, 1)$, $(2, 2)$. How many interesting codes with N digits and containing exactly K digits equal to 1 there exist?

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` o `.pas`.

📎 Among the attachments of this task you will find a template (`interessante.c`, `interessante.cpp`, `interessante.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following function:

C/C++	<code>int conta(int N, int K);</code>
Pascal	<code>function conta(N, K: longint): longint;</code>

Where:

- N is the total number of digits.
- K is the number of digits equal to 1.
- The function shall return the number R of interesting codes with N digits containing K ones, which will be printed to the output file.

Input

File `input.txt` consists of a single line containing integers N, K .

Output

File `output.txt` consists of a single line containing the answer to this problem.

Constraints



- $1 \leq K \leq N \leq 2000$.
- N e K sono tali per cui $NR \leq 7\,000\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.

- **Subtask 1 [10 punti]:** Sample test cases.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [25 punti]:** $N \leq 22$.
- **Subtask 4 [25 punti]:** $N \leq 100$.
- **Subtask 5 [20 punti]:** No limits.

Examples

input.txt	output.txt
1 1	1
8 2	4

Explanation

In the **first sample test case**, the only interesting code considered is 1.

In the **second sample test case**, the corresponding interesting codes are:

```
00010010
00011000
00100100
01001000
```

Filiali bilanciate (filiali)

The *OIS Enterprise* is planning to open F new branches, each of them chosen from a pool of N possible cities. Each of the cities is located beside to the *Sun Highway*, at a distance of K_i , $i = 0, \dots, N$ kilometers from the starting point of the highway. Given a possible choice for the branches, we define its *balancing* has the minimal distance between any two branches chosen (where the distance between two cities is calculated as the difference of the relative K_i, K_j).

What is the maximum possible balancing for a choice of F branches among the N given cities?

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` o `.pas`.

🔗 Among the attachments of this task you will find a template (`filiali.c`, `filiali.cpp`, `filiali.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following function:

C/C++	<code>int apri(int N, int F, int K[]);</code>
Pascal	<code>function apri(N, F: longint; var K: array of longint): longint;</code>

Where:

- N is the number of cities considered.
- F is the number of branches to be opened.
- K is an array indexed from 0 to $N - 1$, containing the kilometers corresponding to each city.
- The function shall return the best possible balancing for a choice of F branches, which will be printed to the output file.

Input

File `input.txt` consists of two lines. Line 1 contains integers N, F . Line 2 contains integers K_i separated by spaces.

Output

File `output.txt` consists of a single line containing the answer to this problem.

Constraints

- $2 \leq F \leq 1000$.
- $1 \leq N \leq 1\,000\,000$.
- $0 \leq K_i \leq K_{i+1} < 2^{31}$ per ogni $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.



- **Subtask 1 [10 punti]**: Sample test cases.
- **Subtask 2 [10 punti]**: $F = 3$.
- **Subtask 3 [10 punti]**: $F = 4$.
- **Subtask 4 [10 punti]**: $F \leq 7$.
- **Subtask 5 [20 punti]**: $N, F \leq 100$.
- **Subtask 6 [20 punti]**: $N \leq 5000$.
- **Subtask 7 [20 punti]**: No limits.

Examples

input.txt	output.txt
5 2 11 27 32 32 53	42
6 3 0 40 60 85 90 100	40

Explanation

In the **first sample test case**, the branches are opened in the first and last city.

In the **second sample test case**, two branches are opened in the first and last city, while the third one can be opened either in the second or third city.

Carta fedeltà (miglia)

Giorgio has just won a grant for travelling to any foreign university. The round trip will be completely reimbursed, providing it does not involve more than K connections. Giorgio now wants to take profit of this trip to gather the highest possible amount of *miles* on his *Alimpiadi* fidelity card.

The *Alimpiadi* company has M unidirectional routes connecting a total of N airports. Each route has a mile value, which could be different to the value of any other route including other ones connecting exactly the same cities. Furthermore, not every airport could be reachable just by taking *Alimpiadi* flights.

What is the maximum amount of miles that Giorgio can gather at most, by taking a round trip made of *exactly* K flights, starting from and arriving to Torino (city number 0)?

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` o `.pas`.

📎 Among the attachments of this task you will find a template (`miglia.c`, `miglia.cpp`, `miglia.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following function:

C/C++	<code>int vola(int K, int N, int M, int da[], int a[], int V[]);</code>
Pascal	<code>function vola(K, N, M: longint; var da, a, V: array of longint): longint;</code>

Where:

- K is the number of flights that Giorgio needs to take.
- N is the number of airports.
- M is the number of *Alimpiadi* routes.
- `da`, `a`, `V` are arrays indexed from 0 to $M - 1$, containing respectively the starting and ending airport, together with the mile value of each route.
- The function shall return the maximum amount of miles that Giorgio can gather, which will be printed to the output file.

Input

File `input.txt` consists of $M + 1$ lines. Line 1 contains integers K , N , M . The subsequent lines contain integers `da[i]`, `a[i]`, `V[i]` separated by spaces.

Output

File `output.txt` consists of a single line containing the answer to this problem.



Constraints

- $2 \leq K \leq 100$.
- $2 \leq N \leq 1000$.
- $2 \leq M \leq 10\,000$.
- $0 \leq v[i] \leq 100\,000$ for all $i = 0 \dots N - 1$.
- $0 \leq da[i], a[i] \leq N - 1$ and $da[i] \neq a[i]$ for all $i = 0 \dots M - 1$.
- There could be different routes connecting exactly the same airports (and possibly with a different mile value).
- There exists at least one round trip starting from airport 0 and consisting of exactly K flights.
- Round trips can use the same route (and airports) more than once.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.

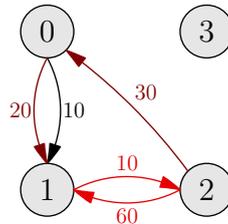
- **Subtask 1 [10 punti]:** Sample test cases.
- **Subtask 2 [20 punti]:** $K \leq 3$.
- **Subtask 3 [20 punti]:** $K \leq 20$ and every airport has *at most 2* outbound flights.
- **Subtask 4 [30 punti]:** $v[i] \leq 1$ for all $i = 0 \dots N - 1$.
- **Subtask 5 [20 punti]:** No limits.

Examples

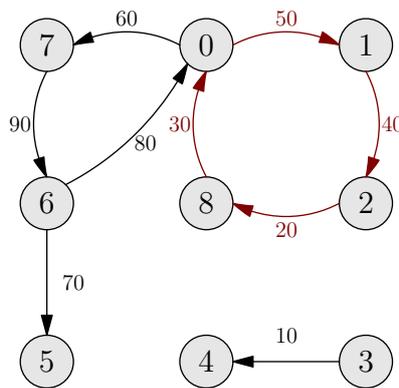
input.txt	output.txt
9 4 5 0 1 10 0 1 20 1 2 10 2 1 60 2 0 30	270
4 9 9 0 1 50 1 2 40 2 8 20 8 0 30 0 7 60 7 6 90 6 0 80 6 5 70 3 4 10	140

Explanation

In the **first sample test case**, are possible both route $0 - 1 - 2 - 0 - 1 - 2 - 0 - 1 - 2 - 0$ and $0 - 1 - 2 - 1 - 2 - 1 - 2 - 1 - 2 - 0$, and the second possibility gives the higher mile value:



In the **second sample test case**, the only possible route is $0 - 1 - 2 - 8 - 0$:

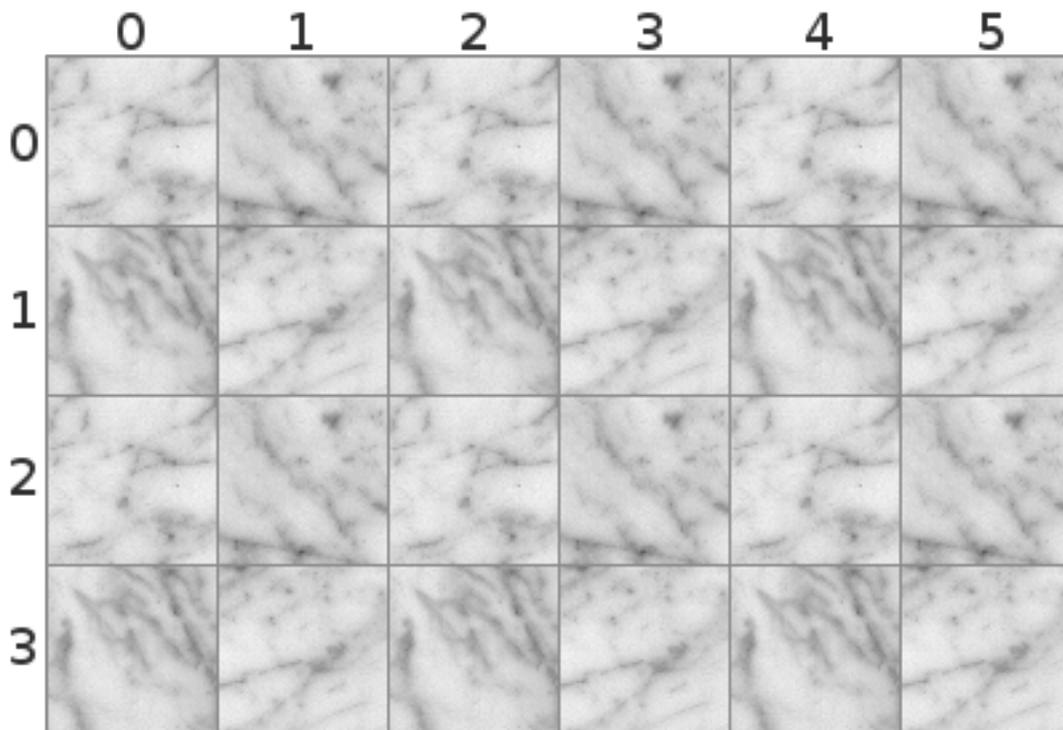


Labigriglia (labigriglia)

The forty-second edition of the annual robotic ant contest is about to begin. This year, Giorgio is in charge of choosing the test that the participating robots will need to pass in order to win.

The previous organizers caused quite a mess last time (who could ever forget the *Anteater arena?*), for this reason Giorgio decided to go with something easier: he will set up a grid in the guise of a labyrinth (the *Labygrid*) and he will reward the ant that will traverse it in the most efficient manner.

A labygrid is nothing more than a pavement formed by $N \times M$ square-shaped tiles. For example, a valid labygrid having $N = 4$ and $M = 6$ can be seen in the following picture:



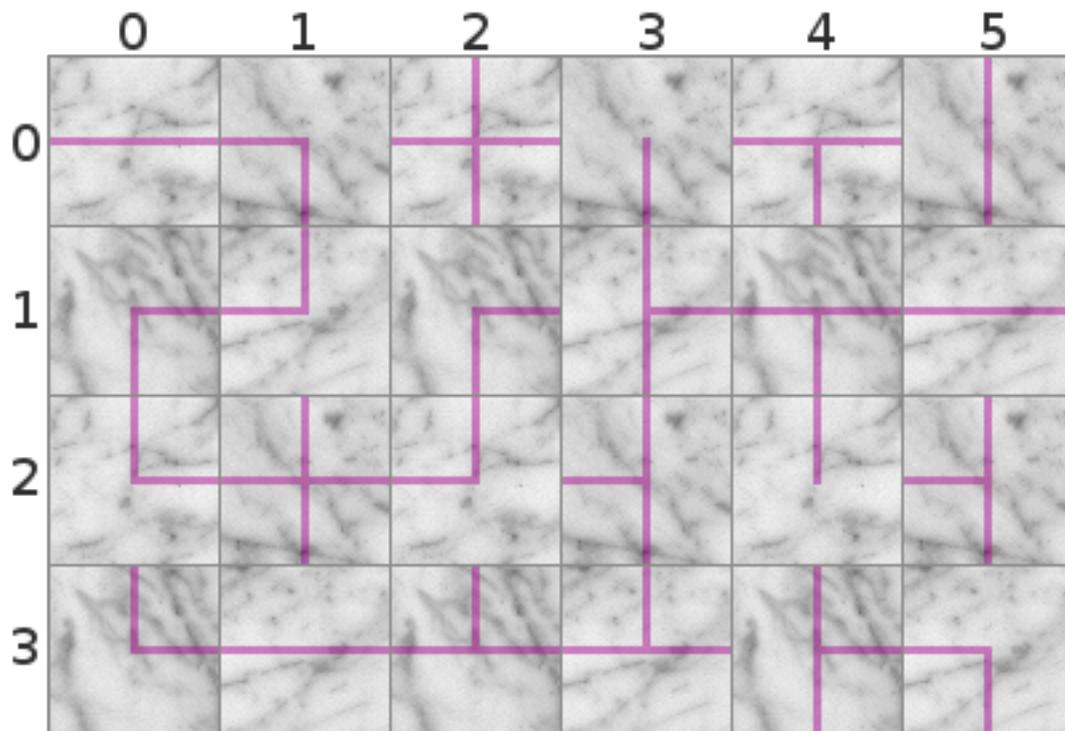
The goal of the robotic ants is to reach the bottom-right corner of the $(N - 1, M - 1)$ tile starting from the top-left corner of the $(0, 0)$ tile. The hard part is to do that by trying to step on as few tiles as possible. The winning ant is the one that will step on the least number of tiles.

Robotic ants can move from a tile to another by the sides or the corners. So, an ant can move to 8 tiles at most (starting from the same tile).

To spice up the game, Giorgio bought an insecticide sprayer (a special kind, for robotic ants) e he’s considering whether to use it or not. Keep in mind that a robotic ant cannot step on the insecticide.

If Giorgio will feel cheerful on the day of the contest, then the insecticide will not be sprayed. However, if Giorgio will feel upset, then he will choose one or more tiles and he will start poisoning them by following this technique: he starts by spraying the poison towards the middle of the tile and, after that, he continues to spray by moving in one of the 4 directions: up, right, down, left.

So, if Giorgio feels especially upset, the “trace” left by the insecticide could reach a point where it resembles a **+** symbol on the tile. For example, in the following labygrid there are two tiles at $(0, 2)$ and at $(2, 1)$ that are really full of insecticide!



To help the contestants understand where the poison is located, Giorgio defines the “poisonous factor” for each tile. At the start, this factor is equal to 0 for all tiles. During the application of the insecticide the factor gets increased as described by this table:

- Spray towards the *up* direction: the poisonous factor of the tile increases by 1;
- Spray towards the *right* direction: the poisonous factor of the tile increases by 2;
- Spray towards the *down* direction: the poisonous factor of the tile increases by 4;
- Spray towards the *left* direction: the poisonous factor of the tile increases by 8.

Nota: this value does not indicate *how much* a tile is poisonous, but *where* it’s poisonous.

👉 In other words, the poisonous factor is a 4-bit binary number. If the high order bit is 1, then the insecticide was sprayed towards the up direction; if the second bit is 1, then the insecticide was sprayed towards the right direction; and so on.

Help Giorgio to organize the upcoming edition of the robotic ant contest! Write a program to determine whether a given labygrid can be solved by a robotic ant and, if that’s the case, compute the *minimum number of tiles* that it will need to step on in order to do that.

Nota: if an ant steps multiple times on the same tile, it will be counted every time!

Implementation

You shall submit exactly one file having extension `.c`, `.cpp` o `.pas`.

📎 Among the attachments of this task you will find a template (`labigriglia.c`, `labigriglia.cpp`, `labigriglia.pas`) with a sample incomplete implementation.

If you use the template, you'll need to implement the following function:

C/C++	<code>int cammina(int N, int M, int griglia[][MAXN]);</code>
Pascal	<code>type matrix = array of array of longint; function cammina(N, M: longint; var griglia: matrix): longint;</code>

Where:

- The N and M integers identify the number of rows and columns of the labygrid, respectively.
- The `griglia` matrix, indexed from 0 to $N - 1$ for the rows and from 0 to $M - 1$ for the columns, describes the labygrid. Specifically, `griglia[i][j]` is the poisonous factor of the (i, j) tile.
- The function must return the minimum number of tiles that a robotic ant must step on in order to solve the labygrid, or the -1 integer, if the labygrid is not solvable.

Input

File `input.txt` consists of $N + 1$ lines. The first line has two integers N and K separated by a space. The next N lines have M integers each, and they describe the `griglia` matrix.

Output

File `output.txt` consists of a single line, containing the -1 integer if a solution does not exist, or a positive integer: the answer to this task.

Constraints

- $1 \leq N, M \leq 1000$.
- $0 \leq \text{griglia}[i][j] \leq 15$ for each i, j .

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain a subtask's score, your program needs to correctly solve all of its test cases.

- **Subtask 1 [10 punti]:** Sample test cases.
- **Subtask 2 [10 punti]:** `griglia[i][j] = 0` for each i, j . No poison!
- **Subtask 3 [35 punti]:** $\min(N, M) = 1$. We should call this a *labycorridor*.
- **Subtask 4 [20 punti]:** $N, M \leq 10$.
- **Subtask 5 [25 punti]:** No limits.

Examples

input.txt	output.txt
<pre> 4 6 10 12 15 4 14 5 6 9 6 7 14 10 3 15 9 13 1 13 3 10 11 11 7 12 </pre>	14
<pre> 4 6 10 12 15 4 14 5 6 9 6 7 14 10 3 15 9 13 1 13 2 10 11 11 7 12 </pre>	13

Explanation

The **first sample test case** is described by the second picture in the task description. A path that steps on 14 tiles in this labygrid is the following:

