Problem A

Find the Strongest Card

Time Limit: 2 seconds

The standard playing cards are of thirteen different ranks, Ace, King, Queen, Jack, and those numbered Ten through Two. A standard deck of cards consists of four each of these ranks, 52 cards in total. In most of the card games, the strengths of the cards are in this order of their ranks, that is, Ace is the strongest and Two is the weakest.

In some of the games popular in Japan and some other regions of East Asia, such as *Daifugo, Zheng Shangyou,* and *Big Two,* the strength order is different. In these games, among the thirteen ranks, Two is the strongest, Ace comes next, followed by King, Queen, and so on, in the ordinary rank order, making Three the weakest. In what follows, this unusual strength order is called *the big two order.*

Given a number of cards, your task is to find the rank of the strongest card among them in the big two order.

Input

The input contains one or more test cases, each in the following format.

$$n$$
$$c_1\ c_2\ \cdots\ c_n$$

Each test case consists of two lines. In the first line, the number of cards, n, is given ($1 \leq n \leq 52$). The second line contains n integers, $c_1, c_2, \ldots, c_n$, which are between 1 and 13, inclusive, representing the ranks of the given n cards. Here, integers 2 through 10 represent ranks Two through Ten, while $1, 13, 12,$ and 11 represent Ace, King, Queen, and Jack, respectively. At most four cards may have the same rank.

The end of the input is indicated by a line containing a zero. The number of test cases does not exceed 100.

Output

For each of the test cases, output in a line the rank of the card strongest in the big two order among the given cards. Here, again, ranks Ace, King, Queen, and Jack, should be represented as $1, 13, 12,$ and 11, and ranks Two through Ten should be represented as their numbers.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	Sample Output 1
4 2 1 10 12 7 3 4 5 4 3 4 5 6 3 11 13 7 8 3 0	2 5 13

Problem B
Vending Machines
Time Limit: 2 seconds

The research facility where you work has a number of rooms and only one straight corridor of length 10^8. Each room has one door leading to the corridor.

You are going to install some vending machines on this corridor. For the convenience of the researchers, you want to place them so that every door has at least one vending machine sufficiently close to it. You are given the positions of the doors and the maximum allowed distance from each door to its nearest vending machine. Find the minimum number of vending machines needed to satisfy this condition.

A vending machine may be installed anywhere on the corridor, possibly at the same position as a door or at an end of the corridor. In this problem, the corridor is regarded as a line segment with no width, and doors and vending machines are treated as points on that line segment.

Input

The input contains one or more test cases, each in the following format.

$$n\ d$$
$$x_1\ x_2\ \cdots\ x_n$$

Each test case consists of two lines. The first line contains two integers n and d, representing the number of doors and the maximum allowed distance from each door to its nearest vending machine, respectively ($1 \le n \le 100$, $1 \le d \le 10^8$). The second line contains the positions of the doors. For each $i = 1, 2, \ldots, n$, x_i is an integer representing the distance from one end of the corridor to the i-th door ($0 \le x_1 < x_2 < \cdots < x_n \le 10^8$).

The end of the input is indicated by a line containing two zeros. The number of test cases does not exceed 100.

Output

For each test case, output the minimum number of vending machines required in a line.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	Sample Output 1
3 5 10 20 40 9 1 0 1 2 3 4 5 6 7 8 2 300 123 724 1 100000000 100000000 0 0	2 3 2 1

In the first test case of Sample Input 1, for example, it is sufficient to install vending machines at positions 15 and 40.

Problem C
Water Remaining
Time Limit: 2 seconds

You made a wooden water tank. When viewed from directly above, the tank is rectangular. The upper edge is at the same height throughout the tank. However, the bottom may not be flat – it could be stepped. Along the length of the tank, the bottom is divided into sections of equal length. Within each section, the bottom is flat and level. The depth may vary from section to section.

The tank was filled with water up to its upper edge. Later, the boards at both ends came off, and water spilled out of the tank. Even so, some water may still remain in the deeper parts of the tank. Calculate the total amount of water remaining. Here, the amount of water kept in one section per unit depth is defined as 1.

Figure C.1 shows the states of the tank. From left to right, they represent the empty state, the fully filled state, and the state in which the boards at both ends have come off, respectively.

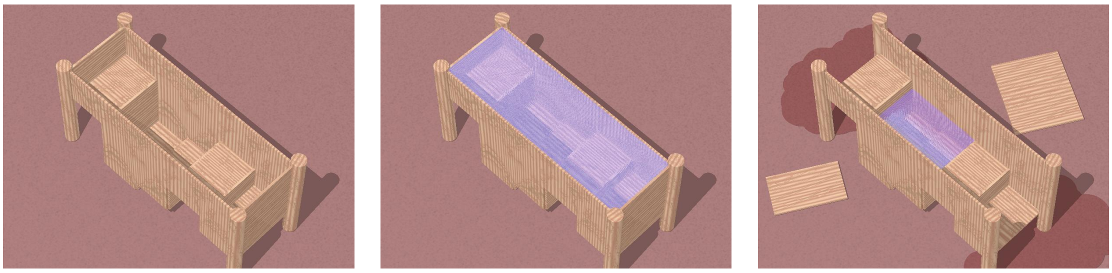

Figure C.1. The tank

In the first test case of Sample Input 1, there are 5 sections whose depths are 3, 8, 7, 4, and 6 respectively. Figure C.2 shows cross-sectional views of this tank. The left figure shows the fully filled state, and the right figure shows the state in which the boards at both ends have come off. The two consecutive sections, the 2nd and 3rd, are deeper than their adjacent sections, so some water remains there, and its surface is at the same height as the bottom of the 4th section. $8 - 4 = 4$ of water remains in the 2nd section, and $7 - 4 = 3$ of water remains in the 3rd section. No water remains in other sections. Thus, the total amount of water remaining is $4 + 3 = 7$.

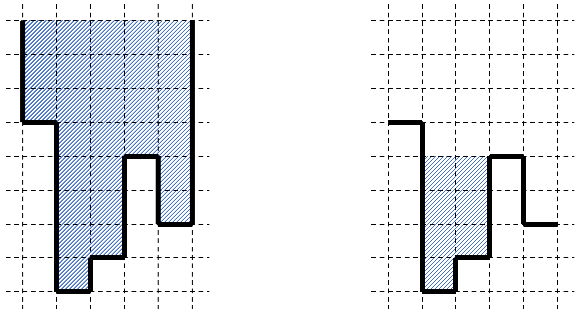

Figure C.2. The cross-sectional views of the first test case of Sample Input 1

Input

The input contains one or more test cases, each in the following format.

$$n$$
$$d_1\ d_2\ \cdots\ d_n$$

A test case consists of two lines. In the first line, the number of sections n is given ($1 \le n \le 100$). For each $k = 1, 2, \ldots, n$, the integer d_k denotes the depth, measured from the upper edge, of the k-th section from one end ($1 \le d_k \le 1000$).

The end of the input is indicated by a line containing a zero. The number of test cases does not exceed 100.

Output

For each test case, output in a line the total amount of water that remains.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	Sample Output 1
5 3 8 7 4 6 9 11 15 14 1 7 5 11 4 9 6 4 6 6 6 4 3 6 5 9 7 8 9 4 1 300 2 300 400 0	7 18 6 13 0 0

Problem D
Frequency Sequence
Time Limit: 2 seconds

Given a positive integer s as its first term, a sequence of positive integers $a_1, a_2, \ldots$ is defined as follows.

- $a_1 = s$
- $a_{i+1} = |\{j \in \{1, 2, \ldots, i\} \mid a_j = a_i\}| \qquad (i \geq 1)$

That is, the $(i+1)$-th term a_{i+1} is the number of times the value of its directly preceding term a_i appears among the first through the i-th terms. For example, when $s = 3$, the first 12 terms of the sequence are 3, 1, 1, 2, 1, 3, 2, 2, 3, 3, 4, and 1.

Given two positive integers s and k, find the value of a_k, the k-th term of the sequence.

Input

The input contains one or more test cases, each in the following format.

s k

A test case consists of two positive integers s and k ($1 \leq s \leq 10^9$, $1 \leq k \leq 10^9$).

The end of the input is indicated by a line containing two zeros. The number of test cases does not exceed 100.

Output

For each test case, output in a line the value of a_k, the k-th term of the sequence.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	Sample Output 1
3 1	3
3 2	1
3 3	1
3 4	2
3 5	1
6 100	12
100000000 100000000	50000000
123456789 987654321	5
31415926 535897932	16621598
0 0	

Problem E
Shopping Master
Time Limit: 2 seconds

In the Icpca Kingdom, two types of currency, coins and gems, are used. To buy a merchandise in shops in this kingdom, you pay either the number of coins shown on its price tag or only 1 gem.

Today, the final day of your sightseeing trip to the Icpca Kingdom, you are visiting a souvenir shop to buy some bottles of local signature liquor. There are n bottles of liquor in this shop. Each liquor is crafted by a different artisan, and therefore the prices may vary. Although you have enough number of coins, you'd like to buy *all* the n bottles with the lowest possible amount of coins, appropriately using the unique benefits provided by the shop.

The detail of the unique benefits is as follows: for some of the bottles in this shop, a complimentary bag of some number of gems are attached to them. Therefore, buying some bottles using the bonus gems that are obtained at prior purchases may reduce the number of coins required.

Starting with no gems, what is the minimum number of coins required to buy all the n bottles?

Figure E.1 describes the prices of the bottles and gems attached to them in the first test case of Sample Input 1. In this case, if you buy the first bottle for 400 coins, you obtain one gem. Then, by buying the third, fourth, and second bottles in this order using gems, you can buy them all without paying any additional coins.

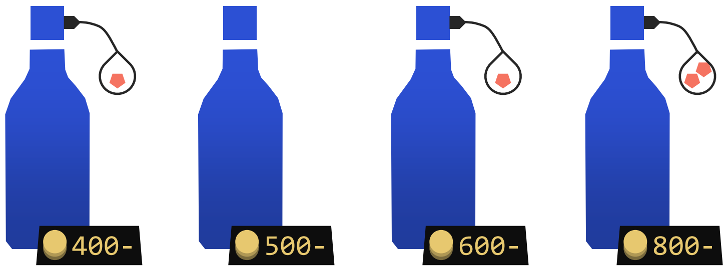

Figure E.1. The first test case of Sample Input 1

Input

The input contains one or more test cases, each in the following format.

n
a_1 b_1
a_2 b_2
$\vdots$
a_n b_n

The integer n ($1 \leq n \leq 10^5$) in the first line is the number of bottles in the souvenir shop. The i-th line of the next n lines describes the information on the i-th bottle ($i = 1, 2, \ldots, n$). The integer a_i ($1 \leq a_i \leq 10^9$) is the number of coins on the price tag of the i-th bottle, and the integer b_i ($0 \leq b_i \leq n$) is the number of gems in the bag attached to the i-th bottle.

The end of the input is indicated by a line containing a zero. The number of test cases does not exceed 2500. The sum of n over all the test cases does not exceed 10^5.

Output

For each test case, output in a line the minimum number of coins required to buy all the n bottles.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1

```
4
400 1
500 0
600 1
800 2
5
1540 0
1430 0
1320 0
1210 0
1100 0
20
861 0
901 0
955 1
602 1
882 1
188 1
817 0
932 2
669 0
621 2
276 0
668 0
825 1
834 1
341 2
545 0
218 0
939 0
179 1
587 1
0
```

Sample Output 1

```
400
6600
1747
```

Problem F
Optimizing a Map Application
Time Limit: 4 seconds

You are optimizing a map application, which is targeted at Icpca City.

The area of Icpca City has a square shape, which is divided into square sections of the same size lined up in n rows and n columns. The section located at the i-th row and the j-th column ($1 \le i \le n, 1 \le j \le n$) is denoted by (i, j). For example, the section at the northwest corner is denoted by $(1, 1)$. Icpca City has its central station occupying a single section, and some number of buildings occupying rectangular areas of one or more sections.

Your map application can calculate the shortest walking time between two specified sections. In the calculation, we assume that people can walk to one of the sections adjacent to north, south, east, and west in 1 minute, and they cannot move diagonally. In addition, we assume that people cannot enter sections occupied by buildings nor step outside of Icpca City. However, they can enter the section of the central station.

During the test phase, you noticed that users frequently ask the walking time from the central station. Write an efficient program that processes many user queries asking for the shortest walking time from the central station to specified sections.

Figure F.1 is a sketch of Icpca City of the first test case of Sample Input 1, showing the shortest path from the central station to the section $(1, 7)$. The central station, the section $(1, 7)$, and buildings are shown in green, cyan stripe, and dark gray, respectively. The shortest walking time is 17 minutes.

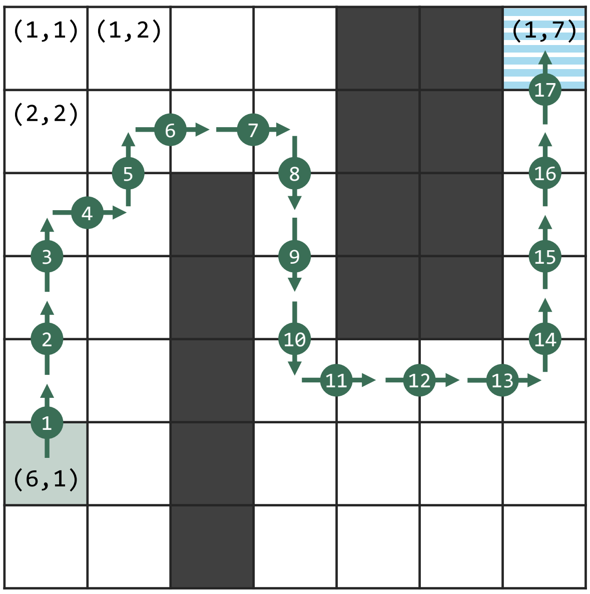

Figure F.1. The first test case of Sample Input 1

Input

The input contains one or more test cases, each in the following format.

$$
\begin{array}{l}
n\ u\ v \\
m \\
a_1\ b_1\ c_1\ d_1 \\
a_2\ b_2\ c_2\ d_2 \\
\vdots \\
a_m\ b_m\ c_m\ d_m \\
q \\
s_1\ t_1 \\
s_2\ t_2 \\
\vdots \\
s_q\ t_q
\end{array}
$$

The first line consists of three integers, n, u, and v ($2 \leq n \leq 10^9, 1 \leq u \leq n, 1 \leq v \leq n$), which mean that the number of sections on each side is n, and that the central station is located at the section (u, v).

The second line contains an integer m ($1 \leq m \leq 200$), representing the number of buildings in Icpca City. The next m lines contain information regarding the areas of the buildings. The i-th line of these m lines consists of four integers, a_i, b_i, c_i, and d_i ($1 \leq a_i \leq b_i \leq n, 1 \leq c_i \leq d_i \leq n$), which mean the i-th building occupies all sections (x, y) that meets $a_i \leq x \leq b_i$ and $c_i \leq y \leq d_i$. It is guaranteed that no sections are occupied by two or more buildings, and the section of the central station is not occupied by any buildings.

The next line contains an integer q ($1 \leq q \leq 2 \times 10^5$), which is the number of queries. The next q lines contain information regarding the user queries. The j-th line of these q lines consists of two integers, s_j and t_j ($1 \leq s_j \leq n, 1 \leq t_j \leq n$), which describe the query to calculate the shortest walking time from the central station to the section (s_j, t_j). It is guaranteed that the section (s_j, t_j) is not occupied by any buildings.

The end of the input is indicated by a line containing three zeros. The number of test cases does not exceed 200. The sum of m over all the test cases does not exceed 200, and the sum of q over all the test cases does not exceed 2×10^5.

Output

For each test case, output q lines. In the j-th line ($1 \leq j \leq q$), output the answer to the j-th query in minutes. If the specified section is unreachable, output `no` as the answer.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	Sample Output 1

<pre>7 6 1
2
3 7 3 3
1 4 5 6
4
1 7
5 5
6 2
6 1
10 1 1
4
5 6 4 4
5 6 7 7
4 4 5 6
7 7 5 6
3
10 10
3 3
5 6
2 1 1
2
1 1 2 2
2 2 1 1
1
2 2
9 1 1
4
2 2 1 8
4 4 2 9
6 6 1 8
8 8 2 9
3
1 5
5 5
9 9
10000000 7777777 123456
1
1000001 9000000 1000001 9000000
3
2525252 9876543
10000000 5252525
123456 123456
0 0 0</pre>
	<pre>17
	11
	1
	0
	18
	4
	no
	no
	4
	24
	48
	17450060
	7351292
	7654321</pre>

Problem G

Avoid Collision

Time Limit: 3 seconds

You have a grid board with n rows and m columns. Let (i, j) denote the cell in the i-th row from the top and the j-th column from the left. Some of the cells on the board may contain obstacles, except for the four corners.

A white piece and a black piece are on the board. Initially, the white piece is placed in the top-left corner cell $(1, 1)$, and the black piece is placed in the bottom-left corner cell $(n, 1)$.

You move the two pieces alternately. You first move the white piece. In one move, you move the white piece to the adjacent cell to the right or below, and you move the black piece to the adjacent cell to the right or above. Two cells are considered adjacent if they share an edge. However, you cannot move a piece onto a cell containing an obstacle. Also, you cannot move a piece onto a cell occupied by the other piece.

You want to move the two pieces $n + m - 2$ times each, so that the white piece reaches the bottom-right corner cell (n, m) and the black piece reaches the top-right corner cell $(1, m)$. Find the number of the pairs of such moving paths modulo $998\,244\,353$.

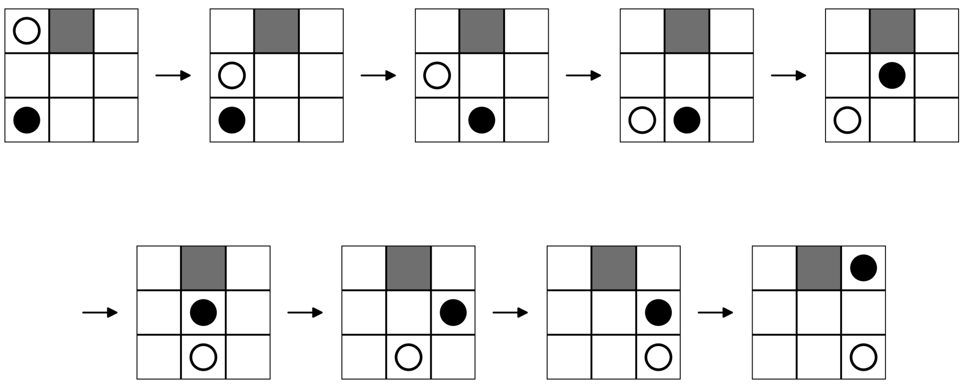

Figure G.1. One way to move the pieces for the first test case of Sample Input 1

Input

The input consists of one or more test cases. Each test case is given in the following format.

$$
\begin{array}{l}
n\ m \\
s_{1,1}s_{1,2}\cdots s_{1,m} \\
\vdots \\
s_{n,1}s_{n,2}\cdots s_{n,m}
\end{array}
$$

The integers n and m denote the numbers of rows and columns of the grid, respectively ($2 \le n \le 10^6, 2 \le m \le 10^6, n \times m \le 2 \times 10^6$). Each of the following n lines contains a string of length m consisting of '`#`' and '`.`', representing whether or not an obstacle is there. If $s_{i,j}$ is '`#`', cell (i, j)

contains an obstacle, and if it is '`.`', the cell does not contain an obstacle. The characters $s_{1,1}$, $s_{1,m}$, $s_{n,1}$, and $s_{n,m}$ are always '`.`'.

The end of the input is indicated by a line containing two zeros. The number of test cases does not exceed 200. The sum of $n \times m$ over all the test cases does not exceed 2×10^6.

Output

For each test case, output in a line the number of the pairs of the moving paths modulo $998\,244\,353$.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1

```
3 3
.#.
...
...
4 7
.......
###.###
###.###
.......
2 2
..
..
11 27
...........................
...........................
...........................
...###...##...###....##....
....#...#..#..#..#..#..#...
....#...#.....###...#......
....#...#..#..#.....#..#...
...###...##...#......##....
...........................
...........................
...........................
0 0
```

Sample Output 1

```
3
0
1
40019649
```

Problem H
Sorting Swim Rings
Time Limit: 2 seconds

You are preparing for the opening of the beach season and sorting colorful swim rings. There are n poles standing vertically from the ground, each with m swim rings threaded on it in a stack. Each swim ring has one of the colors from 1 to n, and every color appears on at least one swim ring. There is also a large area where swim rings can be put temporarily. Initially, this area has no swim ring.

You can perform the operations of moving the swim rings as many times as you like. Each operation is one of the following.

- Take the top swim ring threaded on a pole and thread it onto the top of another pole.
- Take the top swim ring threaded on a pole and put it in the temporary area.
- Choose one swim ring in the temporary area and one of the poles, and thread the swim ring onto the top of that pole.

Since all the swim rings have the same size, the orders of swim rings threaded on the poles never change.

Your goal is to reach a state where all swim rings gathered on each pole have the same color, and the temporary area has no swim ring. You may arbitrarily choose which color of swim rings to gather on which pole. Find the minimum possible number of operations required to reach such a state.

Input

The input contains one or more test cases, each in the following format.

$$\begin{array}{l} n\ m \\ c_{1,1}\ c_{1,2}\ \cdots\ c_{1,m} \\ c_{2,1}\ c_{2,2}\ \cdots\ c_{2,m} \\ \vdots \\ c_{n,1}\ c_{n,2}\ \cdots\ c_{n,m} \end{array}$$

The integer n represents the number of poles, satisfying $2 \le n \le 10$. The integer m represents the number of swim rings initially threaded on each pole, satisfying $1 \le m \le 3 \times 10^4$.

For each i and j ($1 \le i \le n$, $1 \le j \le m$), the integer $c_{i,j}$ represents the color of the j-th swim ring from the bottom on pole i, satisfying $1 \le c_{i,j} \le n$. In particular, the color of the top swim ring on pole i is initially $c_{i,m}$. For each color $1, 2, \ldots, n$, at least one swim ring of that color appears in the test case.

The end of the input is indicated by a line containing two zeros. The sum of n over all the test cases does not exceed 10.

Output

For each test case, output in a line the minimum possible number of operations.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1

```
2 5
2 2 1 1 2
1 1 1 2 2
2 2
1 2
1 1
3 3
1 1 1
1 2 1
3 1 1
0 0
```

Sample Output 1

```
8
3
6
```

In the first test case of Sample Input 1, first put the top swim ring of pole 1 and the top two swim rings of pole 2, all of which have color 2, in the temporary area. Next, thread the top two swim rings of color 1 of pole 1 onto the top of pole 2. Finally, thread the three swim rings of color 2 in the temporary area onto the top of pole 1. Then pole 1 contains only swim rings of color 2, and pole 2 contains only swim rings of color 1. The number of operations is 8, which is the minimum possible.

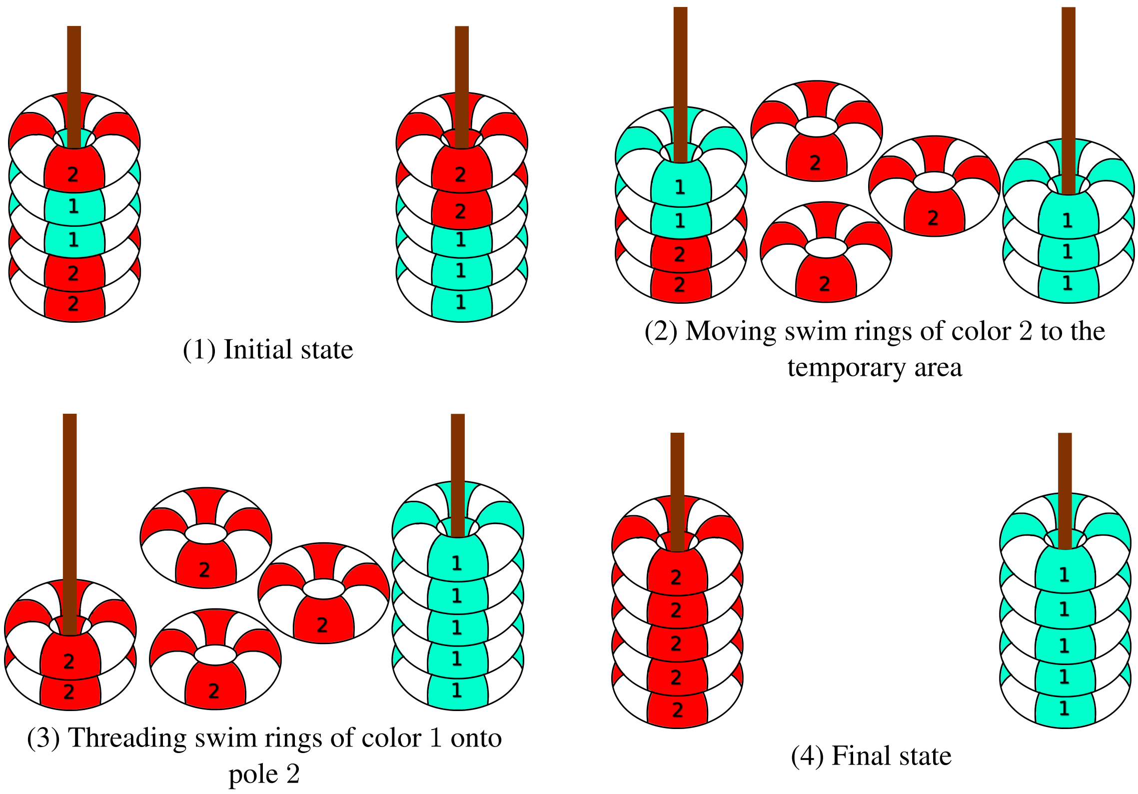

Figure H.1. An example of an optimal sequence of operations in the first test case of Sample Input 1

Problem I
Speed Limit
Time Limit: 4 seconds

A one-way highway extends through the Kingdom of Icpca. The highway is divided from start to end into sections that are 1 km long, and each section has its speed limit.

You are driving a vehicle and want to travel from the start to the end of this highway. The speed of the vehicle can be changed instantaneously at any time. In each change, you choose a *non-negative integer* v and set the speed to v km/h. If the speed before the change is v' km/h, this change incurs a cost of $|v - v'|$. Before leaving the start, the speed of the vehicle is 0 km/h. Also, at the moment the vehicle arrives at the end, the speed must be changed to 0 km/h.

Find the minimum travel time from the start to the end such that the total incurred cost does not exceed the given cost limit and the vehicle does not exceed the speed limits in any sections

Input

The input contains one or more test cases, each in the following format.

n f
a_1 a_2 $\cdots$ a_n

A test case consists of two lines. The first line contains two integers n and f, where n is the number of sections of the highway and f is the limit of the total cost ($1 \le n \le 2 \times 10^5$, $2 \le f \le 10^{10}$). Here, f is even. The second line contains n integers $a_1, a_2, \ldots, a_n$, representing the speed limits of the sections. For $i = 1, 2, \ldots, n$, the speed limit of the i-th section from the start is a_i km/h ($1 \le a_i \le 10^5$).

The end of the input is indicated by a line containing two zeros. The number of test cases does not exceed 10^4. The sum of n over all the test cases does not exceed 2×10^5.

Output

For each test case, output in a line the minimum time in hours required to travel from the start to the end. The output is considered correct if the absolute or relative error does not exceed 10^{-4}.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	**Sample Output 1**
5 120 120 100 40 100 120 5 100 10 20 30 20 10 10 160 30 10 40 10 50 90 20 60 50 30 3 4 2 1 2 3 2 2 1 2 5 20 7 3 8 4 9 15 14 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1 15 60 6 2 3 5 1 7 2 1 5 9 7 5 3 2 6 0 0	0.1050000000 0.3333333333 0.4776190476 2.5000000000 3.0000000000 1.1166666667 5.3285714286 5.4968253968

In the first test case of Sample Input 1, the minimum time can be achieved by changing the speed of the vehicle as follows.

1. At the start, change the speed of the vehicle to 50 km/h. This incurs a cost of $|50 - 0| = 50$.
2. Pass through the first and second sections at 50 km/h. This takes $2/50 = 1/25$ hours.
3. Immediately after passing through the second section, change the speed to 40 km/h. This incurs a cost of $|40 - 50| = 10$.
4. Pass through the third section at 40 km/h. This takes $1/40$ hours.
5. Immediately after passing through the third section, change the speed to 50 km/h. This incurs a cost of $|50 - 40| = 10$.
6. Pass through the fourth and fifth sections at 50 km/h. This takes $2/50 = 1/25$ hours.
7. Upon arriving at the end, change the speed to 0 km/h. This incurs a cost of $|0 - 50| = 50$.

The total cost is $50 + 10 + 10 + 50 = 120$. The total time is $1/25 + 1/40 + 1/25 = 0.105$ hours.

Problem J

Maximum Scaling

Time Limit: 2 seconds

You are given two convex polygons P and Q on the xy-plane. Find the maximum real number s satisfying the following condition.

> Condition: Let Q' be the polygon obtained by multiplying both the x- and y-coordinates of every point of Q by s. It is possible to translate Q' so that its entire boundary and interior are contained in the boundary and interior of P.

Note that the only operation allowed on Q' in the above condition is translation. Rotation and reflection are not allowed.

Figure J.1 (a)–(c) illustrates the three test cases in Sample Input 1. In the first test case, let Q' be the polygon obtained by scaling the polygon Q by 0.5. The red polygon in Figure J.1 (d) shows the polygon obtained by further translating Q' by 5 in the x-direction and by 5 in the y-direction. All points on the boundary and in the interior of this polygon are contained in the boundary and interior of P. The maximum value of s satisfying the condition is 0.5. If s is greater than 0.5, the polygon Q scaled by s cannot be contained in the boundary and interior of P, no matter how it is translated.

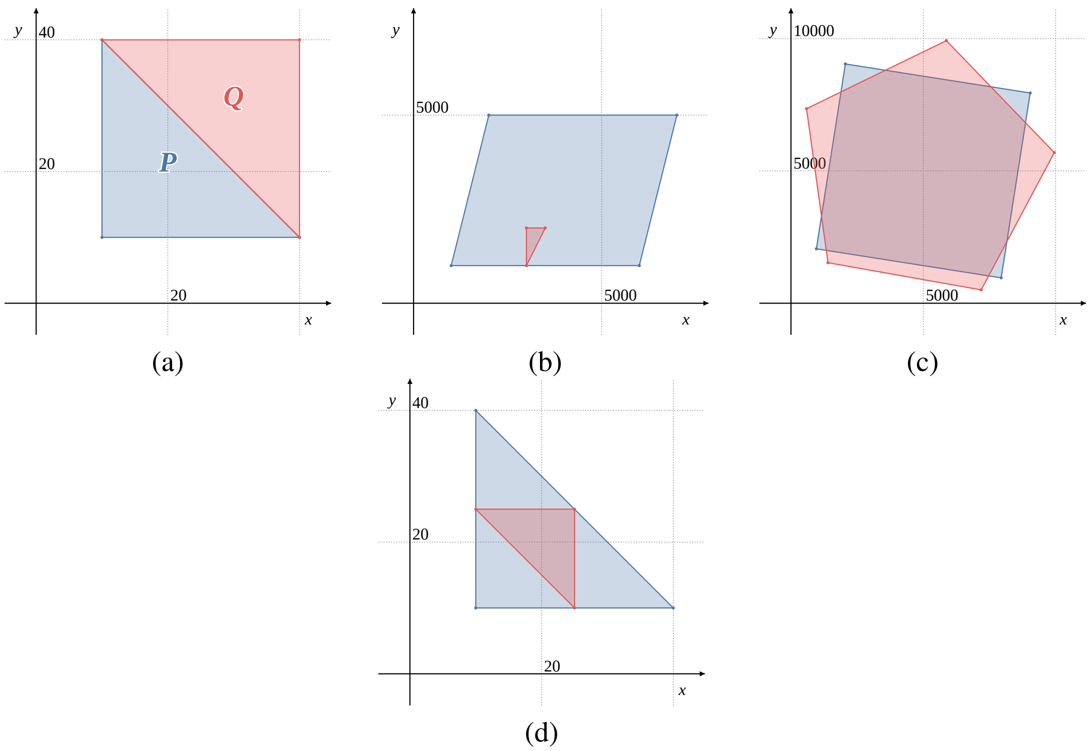

Figure J.1. Illustration of Sample Input 1

Input

The input contains one or more test cases, each in the following format.

$$
\begin{array}{l}
n\ m \\
x_1\ y_1 \\
\vdots \\
x_n\ y_n \\
x'_1\ y'_1 \\
\vdots \\
x'_m\ y'_m
\end{array}
$$

The first line of a test case contains two integers n and m, where n is the number of vertices of the convex polygon P, and m is the number of vertices of Q ($3 \leq n \leq 500$, $3 \leq m \leq 500$). The following n lines contain the coordinates of the vertices of P, $(x_1, y_1), \ldots, (x_n, y_n)$, in counterclockwise order. Similarly, the next m lines contain the coordinates of the vertices of Q, $(x'_1, y'_1), \ldots, (x'_m, y'_m)$, in counterclockwise order. Each coordinate is an integer between 0 and 10^4, inclusive.

It is guaranteed that the polygons P and Q are simple, and that each interior angle is less than 180 degrees.

The end of the input is indicated by a line containing two zeros. The number of test cases does not exceed 100. The sum of n over all the test cases does not exceed 500. The same applies to m.

Output

For each test case, output the maximum value of the real number s in a line. The output is considered correct if the absolute or relative error does not exceed 10^{-4}.

Sample inputs and outputs are available on the Problemset page in DOMjudge.

Sample Input 1	Sample Output 1
3 3 10 10 40 10 10 40 40 40 10 40 40 10 4 3 1000 1000 6000 1000 7000 5000 2000 5000 3000 1000 3500 2000 3000 2000 4 5 2056 9041 959 2056 7944 959 9041 7944 587 7351 1400 1530 7188 504 9952 5692 5872 9923 0 0	0.5 4 0.743571717879