6525 Attacking rooks

Chess inspired problems are a common source of exercises in algorithms classes. Starting with the well known 8-queens problem, several generalizations and variations were made. One of them is the N-rooks problem, which consists of placing N rooks in an N by N chessboard in such a way that they do not attack each other.

Professor Anand presented the N-rooks problem to his students. Since rooks only attack each other when they share a row or column, they soon discovered that the problem can be easily solved by placing the rooks along a main diagonal of the board. So, the professor decided to complicate the problem by adding some pawns to the board. In a board with pawns, two rooks attack each other if and only if they share a row or column and there is no pawn placed between them. Besides, pawns occupy some squares, which gives an additional restriction on which squares the rooks may be placed on.

Given the size of the board and the location of the pawns, tell Professor Anand the maximum number of rooks that can be placed on empty squares such that no two of them attack each other.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N ($1 \leq N \leq 100$) representing the number of rows and columns of the board. Each of the next N lines contains a string of N characters. In the i-th of these strings, the j-th character represents the square in the i-th row and j-th column of the board. The character is either '`.`' (dot) or the uppercase letter '`X`', indicating respectively an empty square or a square containing a pawn.

Output

For each test case, output a line with an integer representing the maximum number of rooks that can be placed on the empty squares of the board without attacking each other.

Sample Input

```
5
X....
X....
..X..
.X...
....X
4
....
.X..
....
....
1
X
```

Sample Output

```
7
```

```
5
0
```

6526 Blogger language

Benjamin's granddaughter Brenda has a blog where she posts articles about school, friends and other life issues. Intrigued by her opinions, Benjamin tried to read it, but very soon he realized it was too hard to read because of Brenda's writing quirks.

Brenda writes without spaces or punctuation marks, and moreover, she uses lower and uppercase letters in a liberal and strange way. For example, one of her posts is "`PrOgRAMmINgiSgrEAt`". Benjamin has trouble noticing the words "`programming`", "`is`" and "`great`" when they are written in this way.

To improve his understanding Benjamin decided to do the following: he will first choose a particular string T and a blog post he is interested in; then he will select a contiguous substring of the post and search for T within the substring, in a case-insensitive way; for each occurrence of T within the substring, he will calculate the number of case mismatches, and finally he will obtain the maximum among all these values. For example, if Benjamin chooses "`GR`" as T and then selects the substring "`PrOgRAM`", he would find a single occurrence "`gR`" for which the number of case mismatches is 1. For the same substring, if "`r`" was chosen as T , he would have found two occurrences, "`r`" with 0 mismatches and "`R`" with 1 mismatch, so the maximum number of mismatches would be 1.

To complicate things further, Brenda included in the blog a script that, after operating with a substring selection, flips the case of all the selected letters. This means that after selecting "`PrOgRAM`" and proceeding as explained above, the sample post would read "`pRoGrammINgiSgrEAt`". If Benjamin selects "`ammINgi`" as a second substring, after calculating his result the post would be left as "`pRoGrAMMinGISgrEAt`", accumulating both flips.

You will be given the string T and the original text of the blog post chosen by Benjamin. You will also be given a list of substring selections Benjamin made, in the order he made them. You need to calculate, for each selection, the maximum number of case mismatches of the occurrences of T in the selected part, considering all the case flips made by previous selections. Notice that the flipping of the case occurs after calculating the result for each selection.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N ($1 \leq N \leq 10^5$) and a non-empty string T of at most 5 letters, representing respectively the number of substring selections and the string to search for. The second line contains a non-empty string P of at most 10^5 letters, indicating the original text of the blog post. Positions of the post are numbered with consecutive integers from left to right, being 1 the leftmost position and $|P|$ the rightmost position. Each of the next N lines describes a substring selection with two integers L and R ($1 \leq L \leq R \leq |P|$) indicating that the substring starts at position L and ends at position R, inclusive.

Output

For each test case, output N lines, each of them containing an integer. In the i-th line write the maximum number of case mismatches of the occurrences of T in the i-th substring selection, considering all the case flips made by previous selections; if no such occurrence exists write the value '`-1`'.

Sample Input

```
3 gR
PrOgRAMmINgiSgrEAt
```

```
1 7
4 18
6 14
9 abCAb
aBcAbCAbaBCAb
1 13
1 13
4 8
5 11
3 11
4 10
1 13
8 8
1 13
```

Sample Output

```
0
2
-1
2
4
1
-1
0
5
2
-1
4
```

6527 Counting ones

Carl is right now the happiest child in the world: he has just learned this morning what the binary system is. He learned, for instance, that the binary representation of a positive integer k is a string $a_n a_{n-1} \ldots a_1 a_0$ where each a_i is a binary digit 0 or 1, starting with $a_n = 1$, and such that $k = \Sigma_{i=0}^{n} a_i \times 2^i$. It is really nice to see him turning decimal numbers into binary numbers, and then adding and even multiplying them.

Caesar is Carl's older brother, and he just can't stand to see his little brother so happy. So he has prepared a challenge: "Look Carl, I have an easy question for you: I will give you two integers A and B, and you have to tell me how many 1's there are in the binary representation of all the integers from A to B, inclusive. Get ready". Carl agreed to the challenge. After a few minutes, he came back with a list of the binary representation of all the integers from 1 to 100. "Caesar, I'm ready". Caesar smiled and said: "Well, let me see, I choose $A = 10^{15}$ and $B = 10^{16}$. Your list will not be useful".

Carl hates loosing to his brother so he needs a better solution fast. Can you help him?

Input

The input file contains several test cases, each of them as described below.

A single line that contains two integers A and B ($1 \leq A \leq B \leq 10^{16}$).

Output

For each test case, output a line with an integer representing the total number of digits 1 in the binary representation of all the integers from A to B, inclusive.

Sample Input

```
1000000000000000 10000000000000000
2 12
9007199254740992 9007199254740992
```

Sample Output

```
239502115812196372
21
1
```

6528 Disjoint water supply

Nlogonia is a queendom that consists of several cities located on a big mountain. The capital city is Logville, located on the mountain peak. Logville has a huge lake with a perfectly round shape, appropriately named "The Big O". This is the only lake with drinkable water in the entire queendom, so it is used to supply all cities. Some cities in Nlogonia are connected with water pipes that allow the distribution of the water. As there are no pumps, each pipe carries water from a city to another city at a lower altitude, using gravity.

Nlogonia's water system has been a source of worries for the Queen, because since cities depend on other cities for their water supply, hot discussions occur about how much water a city is allowed to use. A water supply path is a sequence of cities in decreasing order of altitude, starting in Logville and such that there is a pipe connecting each pair of consecutive cities in the sequence. Two cities have disjoint water supply if and only if there exist two water supply paths, one supply path ending in each of the cities, such that Logville is the only city that is present in both paths. Notice that Logville itself has disjoint water supply with every other city.

The Queen considers disjoint water supply a nice property because it reduces dependency problems and also avoids water outages to spread as quickly through Nlogonia. She therefore ordered a survey to assess the current state of water supply disjointness in the whole queendom. Being the cleverest advisors in the Queen's court, you have been summoned to help calculate the number of pairs of distinct cities that have disjoint water supply.

Input

The input file contains several test cases, each of them as described below.

The first line contains two integers C ($2 \leq C \leq 1000$) and P ($1 \leq P \leq 10^5$), representing respectively the number of cities and the number of water pipes in Nlogonia. Cities are identified with different integers from 1 to C, in strictly decreasing order of altitude (no two cities have the same altitude); Logville is city 1. Each of the next P lines describes a pipe with two integers U and V ($1 \leq U < V \leq C$) indicating that the pipe connects city U with city V. You may assume that no two pipes connect the same pair of cities, and that for each city in Nlogonia there is at least one water supply path that ends in it.

Output

For each test case, output a line with an integer representing the number of pairs of distinct cities that have disjoint water supply.

Sample Input

```
6 6
1 2
1 3
1 4
2 5
2 6
3 6
8 11
1 2
```

```
1 3
1 4
2 5
3 4
6 7
3 6
3 7
4 8
2 6
5 6
```

Sample Output

```
14
26
```

6529 Eleven

In this problem, we refer to the digits of a positive integer as the sequence of digits required to write it in base 10 without leading zeros. For instance, the digits of $N = 2090$ are of course 2, 0, 9 and 0.

Let N be a positive integer. We call a positive integer M an eleven-multiple-anagram of N if and only if (1) the digits of M are a permutation of the digits of N, and (2) M is a multiple of 11. You are required to write a program that given N, calculates the number of its eleven-multiple-anagrams.

As an example, consider again $N = 2090$. The values that meet the first condition above are 2009, 2090, 2900, 9002, 9020 and 9200. Among those, only 2090 and 9020 satisfy the second condition, so the answer for $N = 2090$ is 2.

Input

The input file contains several test cases, each of them as described below.

A single line that contains an integer N ($1 \leq N \leq 10^{100}$).

Output

For each test case, output a line with an integer representing the number of eleven-multiple-anagrams of N . Because this number can be very large, you are required to output the remainder of dividing it by $10^9 + 7$.

Sample Input

```
2090
16510
20140000000000000000000000000
```

Sample Output

```
2
12
0
```

6530 Football

Your favorite football team is playing a charity tournament, which is part of a worldwide fundraising effort to help children with disabilities. As in a normal tournament, three points are awarded to the team winning a match, with no points to the losing team. If the game is drawn, each team receives one point.

Your team played N matches during the first phase of the tournament, which has just finished. Only some teams, the ones with more accumulated points, will advance to the second phase of the tournament. However, as the main objective of the tournament is to raise money, before the set of teams that will pass to the second phase is determined, each team is allowed to buy additional goals. These new goals count as normally scored goals, and may be used to alter the result of any of the matches the team played.

Your team's budget is enough to buy up to G goals. Can you tell the maximum total number of points your team can get after buying the goals, supposing the other teams will not buy any goals?

Input

The input file contains several test cases, each of them as described below.

The first line contains two integers N ($1 \leq N \leq 10^5$) and G ($0 \leq G \leq 10^6$) representing respectively the number of matches your team played and the number of goals your team can buy. Each of the next N lines describes a match result with two integers S and R ($0 \leq S, R \leq 100$), indicating respectively the goals your team scored and received on that match before buying goals.

Output

For each test case, output a line with an integer representing the maximum total number of points your team can get after buying the goals.

Sample Input

```
2 1
1 1
1 1
3 2
1 3
3 1
2 2
4 10
1 1
2 2
1 3
0 4
```

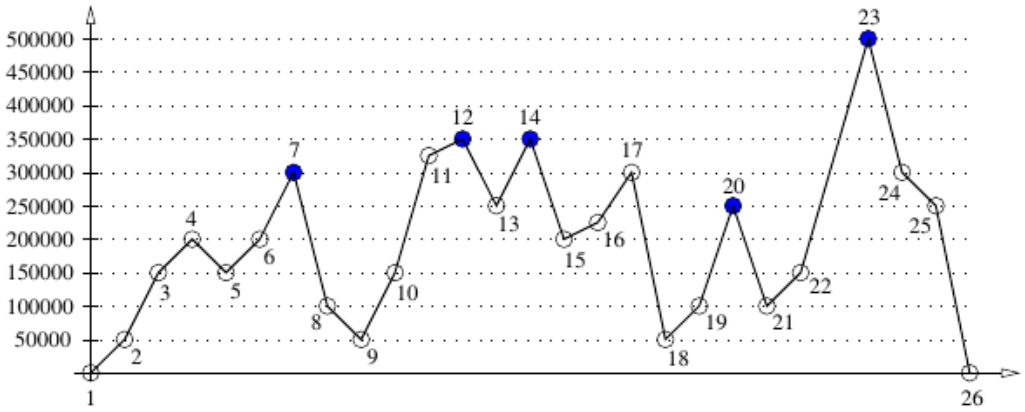
Sample Output

```
4
6
12
```

6531 Go up the Ultras

The topographic prominence of a peak is a measure of special interest to mountain climbers and can be defined as follows: the prominence of a peak p with altitude h, relative to the sea level, is the greatest d such that any path on the terrain from p to any strictly higher peak will pass through a point of altitude $h - d$. If there is no strictly higher peak, then the prominence is h itself. Those peaks with topographic prominence greater than or equal to 150000 centimeters (precision is of great importance to climbers!) have a special name: they are called "Ultras".

You have to write a program that identifies all the Ultras that occur in a two dimensional profile of a mountain range represented as a sequence of points. Note that the horizontal distance between points is not important; all that you need is the altitude of each point. In the picture below, the Ultras are the points 7, 12, 14, 20 and 23.



Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N ($3 \leq N \leq 10^5$) representing the number of points in the profile. The second line contains N integers H_i indicating the altitudes (in centimeters) of the points, in the order in which they appear in the profile ($0 \leq H_i \leq 10^6$ for $i = 1, 2, \ldots, N$). Consecutive points have different altitudes ($H_i \neq H_{i+1}$ for $i = 1, 2, \ldots, N-1$), while the first and the last points are at sea level ($H_1 = H_N = 0$). You may assume that the profile contains at least one Ultra.

Output

For each test case, output a line with the indices of all the Ultras in the mountain range, in the order in which they appear in the profile.

Sample Input

```
5
0 10000 100000 884813 0
7
0 100000 0 200000 180000 200000 0
```

Sample Output

```
4
4 6
```

6532 Hide and seek

In a playground, a group of kids is playing hide and seek. As the name suggests, the game is about kids hiding and seeking other kids. Each kid is either a *hiding* kid or a *seeking* kid. Hiding kids are kids that just try not to be found, while seeking kids are kids that try to find (hiding and seeking) kids.

As you may note, both hiding and seeking kids try not to be found, and for doing this they use some walls that there are in the playground. Each wall is represented by a line segment and each kid is represented by a point in the XY plane. Two kids see each other if and only if the line segment between them does not intersect any wall segment.

Your task is to calculate how many other kids each seeking kid can see. To simplify the problem, you may assume that walls do not intersect even at their endpoints. Moreover, no three points are collinear within the set formed by kids and endpoints of walls; this implies that kids are not inside walls, and that no two kids have the same location.

Input

The input file contains several test cases, each of them as described below.

The first line contains three integers S, K and W representing respectively the number of seeking kids, the total number of kids and the number of walls in the playground ($1 \leq S \leq 10$; $1 \leq K, W \leq 10^4$ and $S \leq K$). Each of the next K lines describes a kid with two integers X and Y ($-10^6 \leq X, Y \leq 10^6$), indicating that the location of the kid in the XY plane is the point (X, Y); the first S of these lines describe seeking kids. Each of the next W lines describes a wall with four integers X_1, Y_1, X_2 and Y_2 ($-10^6 \leq X_1, Y_1, X_2, Y_2 \leq 10^6$), indicating that the two endpoints of the wall in the XY plane are (X_1, Y_1) and (X_2, Y_2). You may assume that wall segments do not intersect and no three points given in the input are collinear.

Output

For each test case, output S lines, each of them containing an integer. In the i-th line write the number of other kids the i-th seeking kid can see.

Sample Input

```
2 3 2
0 0
100 0
0 100
50 -1 48 3
49 49 51 52
4 4 4
-100 0
0 100
0 -100
100 0
3 3 -2 -2
-101 50 101 50
-101 -101 101 -101
-49 -50 49 -50
```

```
5 6 4
40 40
60 10
70 30
60 80
30 81
20 40
0 10 40 50
10 61 30 61
-100 90 200 90
50 20 50 50
```

Sample Output

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

6533 Inverting Huffman

Static Huffman coding is an encoding algorithm used mainly for text compression. Given a text of certain size made of N different characters, the algorithm chooses N codes, one for each different character. The text is compressed using these codes. To choose the codes, the algorithm builds a binary rooted tree having N leaves. For $N \geq 2$ the tree can be built as follows.

1. For each different character in the text build a tree containing just a single node, and assign to it a weight coincident with the number of occurrences of the character within the text.
2. Build a set s containing the above N trees.
3. While s contains more than one tree:
   (a) Choose $t_1 \in s$ with minimum weight and remove it from s.
   (b) Choose $t_2 \in s$ with minimum weight and remove it from s.
   (c) Build a new tree t with t_1 as its left subtree and t_2 as its right subtree, and assign to t the sums of the weights of t_1 and t_2.
   (d) Include t into s.
4. Return the only tree that remains in s.

For the text "`abracadabra`", the tree produced by the above procedure can look like the one on the left of the following picture, where each internal node is labeled with the weight of the subtree rooted at that node. Notice that the obtained tree can also look like the one on the right of the picture, among others, because at steps 3a and 3b the set s may contain several trees with minimum weight.

For each different character in the text, its code depends on the path that exists, in the final tree, from the root to the leaf corresponding to the character. The length of the code is the number of edges in that path (which is coincident with the number of internal nodes in the path). Assuming the tree on the left was built by the algorithm, the code for "`r`" has length 3 while the code for "`d`" has length 4.

Your task is, given the lengths of the N codes chosen by the algorithm, find the minimum size (total number of characters) that the text can have so as the generated codes have those N lengths.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N ($2 \leq N \leq 50$) representing the number of different characters that appear in the text. The second line contains N integers L_i indicating the lengths of the codes chosen by Huffman algorithm for the different characters ($1 \leq L_i \leq 50$ for $i = 1, 2, \ldots, N$). You may assume that there exists at least one tree, built as described, that produces codes with the given lengths.

Output

For each test case, output a line with an integer representing the minimum size (total number of characters) that the text can have so as the generated codes have the given lengths.

Sample Input

```
2
1 1
4
2 2 2 2
10
8 2 4 7 5 1 6 9 3 9
```

Sample Output

```
2
4
89
```

6534 Join two kingdoms

The kingdoms of Nlogonia and Quadradonia fought a long and terrible war that historians have come to call Almost Completely Meaningless (ACM) because nobody can now remember why it started. When the ACM war finally ended, the two kingdoms decided to strengthen their bonds in order to avoid more bloodshed, and for this reason they consulted the International Consortium for the Prevention of Conflicts (ICPC). The ICPC recommended building a single road to connect a city in Nlogonia with a city in Quadradonia, thus allowing commercial and cultural exchange between the two.

Nlogonia and Quadradonia have N and Q cities respectively. The road system of each kingdom consists of a set of bidirectional roads that join pairs of different cities in the same kingdom, such that there is a unique path (i.e. sequence of consecutive roads) that one can take to go from any city in a kingdom to any other city in the same kingdom. The "size" of such a road system is defined as the maximum number of roads that one must take in order to travel between any pair of cities.

Because the ICPC did not specify which two cities should be connected by the new road joining the two kingdoms, the citizens are now worried that the size of the combined road system might be too large. In order to prevent a second ACM war, you would like to convince them that this is not the case, and to this end you need to calculate the expected size of the resulting road system assuming that all possible roads between the two kingdoms are equally likely to be built.

Input

The input file contains several test cases, each of them as described below.

The first line contains two integers N and Q representing the number of cities in each of the two kingdoms ($1 \leq N, Q \leq 4 \times 10^4$). Cities in Nlogonia are identified with different integers from 1 to N , while cities in Quadradonia are identified with different integers from 1 to Q. Each of the next $N-1$ lines describes a road in Nlogonia with two distinct integers A and B indicating that the road connects city A with city B ($1 \leq A, B \leq N$). Each of the next $Q-1$ lines describes a road in Quadradonia with two distinct integers C and D indicating that the road connects city C with city D ($1 \leq C$, $D \leq Q$). The road system of each kingdom is such that there is exactly one path between each pair of cities in the kingdom.

Output

For each test case, output a line with a rational number representing the expected size of the road system after the two kingdoms have been joined, considering that all possible roads connecting them are equally likely to be built. The result must be output as a rational number with exactly three digits after the decimal point, rounded if necessary.

Sample Input

```
4 5
1 2
2 3
4 2
2 3
3 4
4 1
4 5
```

```
1 5
1 2
2 3
3 4
4 5
```

Sample Output

```
5.350
4.400
```