Problem A. Coprime subset

Input file:	`coprime.in`
Output file:	`coprime.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Given N, find the minimal possible value of K such that each K-element subset of any N consecutive positive integers contains at least three integers that are pairwise coprime.

Input

The first and only line of input contains a single integer N ($3 < N \leq 10^7$).

Output

Output the minimal possible value of K.

Example

coprime.in	coprime.out
4	4

Problem B. K-th path

Input file:	`kthpath.in`
Output file:	`kthpath.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Suppose you have a table of N rows and M columns. Each cell of the table contains a single lowercase Latin letter. Consider all paths from the top-left to the bottom-right cell of the table such that from any cell, you are only allowed to move to the neighboring cell to the right and to the bottom. Letters in the cells visited along the path form a string. This string is said to be the *value* of the path.

We take all possible paths and sort them by their values in lexicographic order. Paths with equal values are sorted arbitrarily. Your task is to find the value of the K-th path in this sorted list.

Input

The first line of input contains two integers: the number of rows N and the number of columns M in the given table ($1 \le N, M \le 30$). Each of the next N lines contains exactly M lowercase Latin letters. The last line of input contains a single integer K ($1 \le K \le 10^{18}$). It is guaranteed that the answer exists for given K.

Output

The first and only line of output should contain a single string — the value of K-th path in the sorted list.

Example

kthpath.in	kthpath.out
3 4 abcd efdg hijk 4	abfdgk

Explanation

The values of the ten possible paths are, in lexicographical order: `abcdgk`, `abcdgk`, `abcdjk`, `abfdgk`, `abfdjk`, `abfijk`, `aefdgk`, `aefdjk`, `aefijk`, `aehijk`.

Problem C. Password

Input file:	password.in
Output file:	password.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

What a terrible day — you have lost the password for your e-mail account and do not remember the answer to the secret question! You know the length of the password, and your fingers remember some positions, so you started to use brute force. Suddenly, you have noticed that your e-mail client is very friendly today. It does not simply say "Wrong password", instead it prints: "X symbols of the password are wrong". You have already made two attempts to guess the password. On the first try, you typed A symbols correctly and on the second one, you typed B symbols correctly. Now you are interested in the number of possible passwords that satisfy these conditions. This number can turn out to be very large, so just find it modulo $10^9 + 7$.

Input

First two lines of input contain one string each — your first and second attempts, respectively. Lengths of both strings are equal to some integer N ($1 \le N \le 10^5$; yes, you are crazy on security!). Each string contains only lowercase Latin letters. The last two lines contain numbers A and B, respectively ($0 \le A, B \le N$).

Output

Output should contain the number of possible passwords modulo $10^9 + 7$.

Example

password.in	password.out
ab ac 1 1	24

Explanation

Possible passwords are: aa, ad, ae, ..., az.

Problem D. Permutations

Input file:	`permutations.in`
Output file:	`permutations.out`
Time limit:	10 seconds
Memory limit:	256 Mebibytes

Consider a cyclic alphabet which consists of the first ten letters of the usual Latin alphabet. It is called cyclic because the next letter after '`a`' is '`b`', the next after '`b`' is '`c`' and so on. The last letter '`j`' is followed by the first letter '`a`'.

You are given an initial string S which contains only the letters of this cyclic alphabet. You should process queries of three kinds:

- Reverse substring of S from L to R, inclusive.
- For substring of S from L to R, inclusive, replace each character with the D-th next character in the cyclic alphabetical order.
- For substring of S from L to R, inclusive, return the number of distinct permutations of the characters of this substring modulo $10^9 + 7$.

Input

The first line of input contains one integer N ($1 \le N \le 10^5$) — the length of the string S, followed by the string S on the second line. The string contains only lowercase Latin letters from '`a`' to '`j`', inclusive. The third line contains an integer M ($1 \le M \le 10^5$) — the number of queries. It is followed by M lines, each of which represents one of the following three types of queries:

- `-1 L R` ($1 \le L \le R \le N$) — reverse substring from L to R.
- `0 L R D` ($1 \le L \le R \le N$, $0 < D \le N$) — replace each letter with the D-th next letter in the cyclic alphabetical order.
- `1 L R` ($1 \le L \le R \le N$) — return the number of distinct permutations of the characters of substring $[L, R]$ of S.

Output

For each query of type "`1 L R`", return the answer modulo $10^9 + 7$.

Example

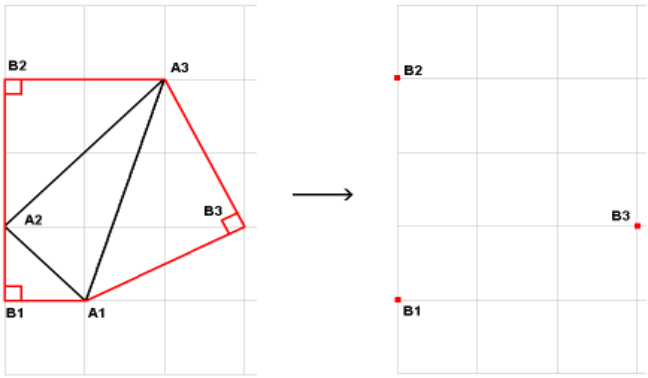
permutations.in	permutations.out
6 abcabc 3 -1 1 6 0 1 3 1 1 1 6	180

Problem E. Polygon

Input file:	`polygon.in`
Output file:	`polygon.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

There is a convex polygon $A_1A_2A_3 \ldots A_N$ on the plane. First, we construct N triangles $\triangle A_1B_1A_2$, $\triangle A_2B_2A_3$, ..., $\triangle A_NB_NA_1$: each triangle $\triangle A_iB_iA_j$ where $j = i \bmod n + 1$ is a right isosceles triangle that lies outside the polygon and has hypotenuse A_iA_j. After that, we erase the polygon and the triangle edges, so that only points B_i remain on the plane.

Your task is, given these points B_i, to restore the original polygon.



Input

The first line of input contains an integer N ($3 \le N \le 10^4$). The next N lines contain coordinates of points B_i (one pair of coordinates per line) in clockwise order starting with B_1. Line number $(i + 1)$ contains coordinates of B_i with exactly four digits after decimal point. Coordinates don't exceed 10^5 by absolute value.

Output

If the solution is unique, print the number of polygon's vertices on the first line of output, then output N lines with coordinates of all vertices of the original polygon in clockwise order starting with A_1. The coordinates in your output should be within 10^{-5} of their exact values. If there is more than one solution, output just a single integer "`-1`" instead; if there is no solution, output "`-2`".

Example

polygon.in	polygon.out
3 0.0000 1.0000 0.0000 4.0000 3.0000 2.0000	3 1.00000 1.00000 0.00000 2.00000 2.00000 4.00000

Problem F. Mask Matching

Input file:	re.in
Output file:	re.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

In this problem, we consider strings consisting of lowercase Latin letters. A *mask* is a string composed of lowercase Latin letters and template symbols '*' and '?'. Each symbol '*' corresponds to any (possibly empty) sequence of letters, and each symbol '?' corresponds to any single letter. A string S *matches* a mask T if it is possible to replace all template symbols in T by some corresponding sequences so that the resulting string is equal to S.

You are given two masks R_1 and R_2. Consider all possible pairs of strings A and B such that A is matched to R_1 and B is matched to R_2. For each such pair, find their longest common subsequence C (recall that a *subsequence* of a string is a string obtained from the original one by erasing zero or more symbols). From all possible values of C, find the longest one. If there are several longest values of C, find the lexicographically first one and output it. If there are infinitely many different values of C, output "%)".

Input

The first two lines of input contain two nonempty strings composed of lowercase Latin letters and signs '*' and '?'. Their lengths will not exceed 1000.

Output

On the first line of output, write the required string.

Examples

re.in	re.out
* *	%)
ab b?	a

Problem G. Skating system (experimental)

Input file:	`skating.in`
Output file:	`skating.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Warning! Some of the judges had troubles understanding this problem. The statement appears to be written in some strange language. Read it at your own risk! Good luck!

You may heard about such unusual type of sport like dancesport. If say shortly, there are couples, they dance dances (usually 5, sometimes 10) and there are, of course, judges (always an odd number), which score dances, danced by couples. Because the number of couples could be very large, there maybe several rounds in a competition. The last one called "final round" and there maybe at most 8 couples in it. All other rounds are "preliminary". The scoring is regulated by so called "The Skating System". In this task you will automate judging of the one dance competition.

Here are the rules of The Skating System.

Rule 1. In all preliminary rounds, judges must call back the number of couples requested by the Chairman of Adjudicators.

The couples that advance to the next round are simply those with the most callback marks. Their number must equal to the number of requested couples. For simplicity, if there is a tie on the border of advancing couples, then couples with lower numbers are advanced. Couples that have not been advanced to the next round take place according to the number of their marks. Couples with equal number of marks share their places.

Rules 2, 3, 4 apply specifically to the competition judges.

Rule 2. In a final round all couples must receive a placement from each judge.

Rule 3. In a final round a judges first choice is marked "1", second choice is marked "2", third choice is marked "3", and so on.

Rule 4. In a final round a judge may not tie any couple for any place of any dance

Rules 5, 6, 7, and 8 apply to tabulating the results for the individual dances in a section or for a single-dance section.

Rule 5. How to allocate positions in each dance

The Skating System is based on the marks a couple receives from a majority of judges. The first and simplest step is to ascertain what makes up a majority. The next step is to place the winner by inspecting the marks for the number of 1st places. It is important to note that in this rule we simply count the number of places, we do not add them together.

The couple who has received the majority of 1st place marks is the winner of that dance and their marks have no further impact on the tabulation process. The next step is to determine who is to be placed second. This follows a similar process. In this case, however, we count the number of "2nd place and higher marks" for the remaining couples. The next step is to determine who is to be placed third. We, similarly, count the number of "3rd place and higher marks" for each of the remaining couples. This process is repeated until all couples have been placed.

Rule 6. More than one couple have a majority for the same place.

Rule 6 is a simple follow-on to Rule 5. The position is allocated according to which couple has the greater majority. All couples with the majority are placed before you consider the remaining couples.

Rule 7. If two or more couples have an equal majority for the same position

Now is the time to add together the place-marks and not just count them. Add (not count!) the place-marks for each couple that has the majority. The couple with the lowest total (sum) is awarded the position. This process continues until all couples with the equal majority have been awarded positions.

OK!

Next question, "Same majority, same sum...?" The last part of Rule 7 defines a tie. There are situations where no matter how many rules you apply you cannot separate the couples. In the event of two couples having an equal majority and also an equal sum, we go to the next order of scores, FOR THESE COUPLES ONLY. If the next order still gives us an equal majority and sum we go to the next one, and the next until we reach the last possible. For 6 couples this will be "6th and higher," for 7 couples "7th and higher," and for 8 couples "8th and higher." Remember that if you have more than 8 couples you will not be running a final! If we still have a tie at the last order of scores then each couple is awarded the highest of the positions that we are working with. For example, if we have two couples and we are looking to place 3rd and 4th each couple will be awarded the 3rd position. Note, that in this case there no couple will be on the 4th place.

Rule 8. If no couple receives a majority for the position under review.

After Rule 7, Rule 8 is simplicity itself. If no couple achieves a majority of 1st place marks then you move on to the "2nd place and higher". If there is no majority there you continue onto the next and the next until one or more couples achieve a majority.

Some simplifications:

- for each judge you are given his preferences — the list of couples in order from the best to the worst (in opinion of this judge). Unlike in reality, judge's opinion will not change between rounds;
- if there are N couples in a preliminary round, judges must mark $(N + 1)/2$ couples;
- the final round is the round with 8 or less couples.

Now, given judges preferences, output final standings.

Input

The first line of the input file contains two integer numbers N — the number of couples and M — the number of judges ($1 \le N \le 1000$, $1 \le M \le 13$, M is odd). Each of the next M lines contains N distinct integers — couple numbers in the order of preference of this judge, from the most preferred to the least one. Couple number is a positive integer not larger than 10000 that uniquely identifies the couple. Each couple number will be listed in every judge's preference list exactly once.

Output

Output N lines, containing two integers each — the number of couple and its place, separated with a space. Output couples in order from highest place to lowest. In case when two couples share a place output them in increasing order of their numbers.

Example

skating.in	skating.out
15 5 115 111 118 116 124 117 112 114 113 119 120 121 122 123 125 115 116 113 114 111 118 117 112 121 122 119 120 123 124 125 112 113 114 116 115 122 123 111 118 117 119 124 120 121 125 116 115 114 118 112 111 117 113 119 123 120 121 122 124 125 116 114 117 111 118 115 112 113 119 123 121 122 124 120 125	116 1 115 2 114 3 111 4 118 5 112 6 117 7 113 8 122 9 123 9 124 9 119 12 120 12 121 12 125 12

The final consists of 8 couples. According to preferences list couples will have following marks in the final round:

111	2	5	6	6	4
112	6	8	1	5	7
113	8	3	2	8	8
114	7	4	3	3	2
115	1	1	5	2	6
116	4	2	4	1	1
117	5	7	8	7	3
118	3	6	7	4	5

There are five judges for the dances. The majority is therefore 3.

1. There is no majority of 1st places so move to the "2nd and higher".

2. 115 and 116 both have the same total of 3 "2nd and higher" marks, which is a majority. The sum of these 3 marks is also equal (4). To break the tie we must move to the "3rd and higher".

3. 115 and 116 still have the same tie as neither achieved any 3rd place marks from the judges. Counting the "4th and higher" we find that 115 has 3 "4th and higher" marks and 116 has 5. Both of these are a majority. By virtue of the larger majority 116 is awarded 1st place and, being only a two-couple tie, 115 is awarded 2nd place.

4. We must now go back to the "3rd and higher" for the remaining couples. 114 has a majority of 3 "3rd and higher" place marks and is given 3rd place.

5. Moving onto the "4th and higher" for the remaining couples we find that none of them has a majority, so we move straight on to the "5th and higher".

6. Counting "5th and higher" we find that 111 and 118 both have a majority of 3 "5th and higher" place marks. The sum of these 3 place marks is lower for 111 (11) than for 118 (12). As a result of the lower sum 111 is given 4th place and 118 gets 5th place.

7. We now inspect the "6th and higher" for the remaining three couples. 112 has a majority of 3 place marks and is placed 6th.

8. In like manner 117 has a majority of 4 "7th and higher" place marks and is given 7th place.

Problem H. Square

Input file:	`square.in`
Output file:	`square.out`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

There is a square on the plane with sides parallel to coordinate axes and vertices at integer points. A random convex polygon with vertices at integer points was inscribed into that square. More specifically, an *inscribed* polygon is a strictly convex non-degenerate polygon with vertices at integer points which has at least one vertex on each side of the square. All such polygons have equal probability of being chosen.

Your task is to find the expected area of such polygon.

The area always can be represented as an irreducible fraction A/B. You should calculate the number which is equal to A/B in modular arithmetic modulo P. Recall that $B^{-1} \bmod P$ is an integer C, $0 \le C < P$ such that $(B \cdot C) \bmod P = 1$. It's guaranteed that $\gcd(B, P) = 1$.

Input

The first line of input contains two integers N and P where N is the size of the square and P is the modulus ($1 \le N \le 40$, $2 \le P \le 2 \cdot 10^9$).

Output

On the first line of output, write one integer — the answer to the problem.

Examples

square.in	square.out
1 101	41
2 101	78

Explanation

In the first example, the expected area is 3/5: the five possible polygons are four triangles with area 1/2 and one square with area 1. So $A = 3$ and $B = 5$. We first find that modulo $P = 101$, we have $C = B^{-1} = 81$. Now, $A/B = A \cdot C = 243 = 41 \mod 101$.

Problem I. Strings

Input file:	`strings.in`
Output file:	`strings.out`
Time limit:	3 seconds
Memory limit:	256 Mebibytes

You are given two strings. You are allowed to remove some characters from them, but the number of consecutive removed characters must not exceed W. Your task is to make strings equal by removing the minimal possible total number of characters. String comparison is case sensitive.

Input

The first line of input contains one integer W — the maximal number of consecutive removed characters ($1 \le W \le 1500$). The second and the third lines contain the two given strings. Strings consist of digits and uppercase and lowercase Latin letters. It is guaranteed that the length of each string is positive and does not exceed 1500.

Output

The output should contain the resulting string. If there are several answers, output any of them. If there is no answer, output "`No solution`".

Examples

strings.in	strings.out
1 xabcd aefdz	No solution
2 xabcd aefdz	ad

Problem J. Subsequence

Input file:	`subsequence.in`
Output file:	`subsequence.out`
Time limit:	3 seconds
Memory limit:	256 Mebibytes

You are sequence A of length N. A *subsequence* is any sequence that can be obtained from A by removing zero or more elements. Let's take all possible subsequences of A, remove duplicate subsequences, sort the rest lexicographically and denote the resulting list as B. You should process two types of queries:

1. What number will sequence X have in B? Note that the answer can be quite large, so you should find this number modulo $10^9 + 7$.
2. What is the Y-th sequence in this list? Sequences are numbered starting from 1.

Input

The first line of input contains two integers N and M where N is the number of elements in A and M is the number of queries ($1 \le N \le 5 \cdot 10^5$, $1 \le M \le 100$). The second line contains N integers separated by spaces — elements of A. Each element is a nonnegative integer less than 20. Each of the next M lines contains a description of a query. The first number in a description is its type: 1 or 2. If query type is 1, the rest of the line contains sequence X. If query type is 2, there is a second number Y on the same line. It is guaranteed that each subsequence X is an element of B, each Y does not exceed the number of sequences in A and $1 \le Y \le 10^{18}$.

Output

For each query, write the answer to it on a separate line. For a query of type 1, output the number of the given subsequence modulo $10^9 + 7$. For a query of type 2, output the requested subsequence separating its elements by spaces.

Examples

subsequence.in	subsequence.out
5 2 1 2 3 1 2 1 3 2 2 2	27 1 1

Explanation

The sorted list of subsequences of sequence $A = \{1, 2, 3, 1, 2\}$ is the following:

1. 1
2. 1, 1
3. 1, 1, 2
4. 1, 2
5. 1, 2, 1
6. 1, 2, 1, 2
7. 1, 2, 2
8. 1, 2, 3
9. 1, 2, 3, 1
10. 1, 2, 3, 1, 2
11. 1, 2, 3, 2
12. 1, 3
13. 1, 3, 1
14. 1, 3, 1, 2
15. 1, 3, 2
16. 2
17. 2, 1
18. 2, 1, 2
19. 2, 2
20. 2, 3
21. 2, 3, 1
22. 2, 3, 1, 2
23. 2, 3, 2
24. 3
25. 3, 1
26. 3, 1, 2
27. 3, 2