Problem A. Number of paths in acyclic graph

Input file:	`countpaths.in`
Output file:	`countpaths.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given an oriented acyclic graph. Find the number of distinct paths from vertex 1 to vertex n.

Input

First line of the input file contains two integer numbers n and m — number of vertices and edges in the graph ($2 \le n \le 10^5$, $2 \le m \le 2 \cdot 10^5$).

Each of the following m lines contains two integer numbers: numbers of vertices connected by the corresponding edge.

Output

Output the number of distinct paths from vertex 1 to vertex n taken modulo $10^9 + 7$.

Example

countpaths.in	countpaths.out
4 4 1 2 1 3 3 2 2 4	2

Problem B. Knapsack

Input file:	`knapsack.in`
Output file:	`knapsack.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given n items with masses $m_1, \ldots, m_n$ and costs $c_1, \ldots, c_n$ respectively.

The knapsack can hold items with total mass up to m. You are to find the set of items having the maximal possible total cost which can be held by the knapsack.

Input

First line of the input file contains a positive integer number n not exceeding 1000 and positive integer number m not exceeding 10000.

Second line contains n positive integer numbers m_i, each of them does not exceed 100.

Third line contains n positive integer numbers c_i, each of them does not exceed 100.

Output

Output the number of items in the set in the first line.

Second line must contain numbers of these items (ranging from 1 to n).

Example

knapsack.in	knapsack.out
4 6 2 4 1 2 7 2 5 1	3 1 3 4

Problem C. Longest common subsequence

Input file:	`lcs.in`
Output file:	`lcs.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given two sequences. Find their longest common subsequence.

Input

First line of the input file contains an integer number N — length of the first sequence ($1 \le N \le 2000$). Second line contains N numbers — the first sequence. All sequence elements do not exceed 10^9 by absolute value.

Third line of the input file contains an integer number M — length of the first sequence ($1 \le M \le 2000$). Fourth line contains M numbers — the second sequence. All sequence elements do not exceed 10^9 by absolute value.

Output

Output the length of the longest common subsequence in the first line. Output the subsequence in the second line. If there are several longest common subsequnces — output any of them.

Example

lcs.in	lcs.out
3 1 2 3 4 2 3 1 5	2 2 3

Problem D. Levenshtein distance

Input file:	`levenshtein.in`
Output file:	`levenshtein.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider a string and a set of operations:

- Substitute one character of the string.
- Delete one character from any position of the string.
- Insert one character in any position of the string.

E.g., using first operation one can transform “ABC” to “ADC”, using second — to “AC”, using third — to “ACBC”.

Minimal number of such operations needed to transform one string to the other is called the *Levenshtein distance*.

You are given two strings. Find the Levenshtein distance between them.

Input

Input file contains two lines each of the containid one of the given strings. Lengths of these strings do not exceed 5000 and strings consist only from capital Latin letters.

Output

Output one number — the Levenshtein distance.

Example

levenshtein.in	levenshtein.out
ABCDEFGH ACDEXGIH	3

Problem E. Longest increasing subsequence

Input file:	lis.in
Output file:	lis.out
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given a sequence of numbers. Find its longest increasing subsequence.

Input

First line of the input file contains an integer number N — length of the sequence ($1 \le N \le 2000$). Second line contains N numbers — the sequence. All sequence elements do not exceed 10^9 by absolute value.

Output

Output the length of the longest increasing subsequence in the first line. Output the subsequence in the second line. If there are several longest increasing subsequnces — output any of them.

Example

lis.in	lis.out
6 3 29 5 5 28 6	3 3 5 28

Problem F. Maximal weight matching in tree

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

Matching is a subset of graph edges such that no two edges in it share a vertex

You are given a weighted tree. You are to find the maximum weight matching in it.

Input

First line contains an integer number n — number of vertices in tree ($2 \le n \le 10^5$).

Each of the following $(n-1)$ lines describes an edge and contains three numbers: numbers of vertices connected by this edge and the weight w_i ($1 \le w_i \le 10^9$).

Output

Output one number — the weight of maximal weight matching in the given tree.

Example

matching.in	matching.out
4 1 2 10 1 3 1 3 4 1	11

Problem G. Matrix multiplication

Input file:	matrix.in
Output file:	matrix.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Matrix product is said to be fully parenthized if one of the following holds:

- It consists of one matrix.
- It is a put in parentheses product of two fully parenthized products.

A full parenthezation is called optimal if the number of operations needed to calculate the product is minimal possible.

You are to find the optimal full parenthezation for matrix product.

Input

First line of the input file contains an integer number n — number of matrices ($1 \le n \le 400$).

Each of the following n lines contains two integer numbers a_i and b_i — number of rows and columns in the i-th matrix, respectively ($1 \le a_i, b_i \le 100$).

It is guaranteed that $b_i = a_{i+1}$ for each $1 \le i \le n-1$

Output

Output the optimal parenthezation. If there are several of them, output any.

Example

matrix.in	matrix.out
3 10 50 50 90 90 20	((AA)A)

Note

In this case there are two possible parenthezations: $((AA)A)$ $(A(AA))$. The number of operations is $10{\cdot}50{\cdot}90{+}10{\cdot}90{\cdot}20 = 63000$ for the first one, and $10 \cdot 50 \cdot 20 + 50 \cdot 90 \cdot 20 = 100000$ for the second one.

Problem H. Longest subpalindrome

Input file:	palindrome.in
Output file:	palindrome.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Palindrome is a string which reads the same from both directions.

Subpalindrome is a sequence of characters from the string (not necessarily consecutive) which is a palindrome. E.g., "HELOLEH" is a subpalindrome of "HTEOLFEOLEH".

You are to find the longest subpalindrome of the given string.

Input

Input file contains a string of capital Latin letters. Its length do not exceed 2000.

Output

Output the length of the longest subpalindrome on the first line. Second line must contain the maximal subpalindrome itself. If there are several longest subpalindromes, you can output any of them.

Example

palindrome.in	palindrome.out
HTEOLFEOLEH	7 HEOLOEH

Problem I. Travelling salesman problem

Input file:	salesman.in
Output file:	salesman.out
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given an non-oriented weighted graph without loops and parallel edges. You are to find the path visiting all its vertices and having the minimal weight.

Input

First line contains two integer numbers n and m — number of vertices and edges in the graph ($1 \le n \le 18$, $0 \le m \le \frac{n \cdot (n-1)}{2}$). Each of the following m lines describes one edge and contains three numbers: numbers of vertices connected by the edge and the weight of the edge ($1 \le a_i, b_i \le n$, $1 \le w_i \le 10^8$).

Output

Output one number — weight of the path. If there no such path, output -1.

Examples

salesman.in	salesman.out
4 6 1 2 20 1 3 42 1 4 35 2 3 30 2 4 34 3 4 12	62
4 3 1 2 1 1 3 1 1 4 1	-1

Problem J. Brackets Subsequences

Input file:	brackets.in
Output file:	brackets.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider bracket sequences with one type of brackets. Given a sequence of brackets, your task is to find the number of its different subsequences that are regular brackets sequences.

For example, the sequence "(((()))(" has 8 such subsequences: "((()))", "(())()", "((()))", "(()())", "(())", "()()", "()", and "".

Input

The input file contains a sequence of brackets. The sequence is not empty, its length does not exceed 300.

Output

Output the number of its different subsequences that are regular brackets sequences.

Examples

brackets.in	brackets.out
(((()))(	8

Problem K. String Decomposition

Input file:	decomp.in
Output file:	decomp.out
Time limit:	2 seconds
Memory limit:	256 megabytes

For a string α and an integer n define α^n as the concatenation of n copies of α. For example, $\texttt{aab}^4 = \texttt{aabaabaabaab}$.

Each string S can be decomposed as $S = S_1^{d_1} S_2^{d_2} \dots S_k^{d_k}$. There can be several ways to make such decomposition. The weight of the decomposition is the sum $|S_1| + |S_2| + \dots + |S_k|$ where $|Z|$ is the length of the string Z.

Given S find its decomposition which has the minimal possible weight.

Input

The input file contains the string S. S contains only capital letters of the English alphabet, its length doesn't exceed 5 000.

Output

The first line of the output file must contain w — the minimal possible weight of the decomposition of S. Let k be the number of elements in the optimal decomposition. The following k lines must contain two elements each — S_i and d_i separated by a space.

If there are several optimal decompositions, describe any one.

Example

decomp.in	decomp.out
ABABAAABABA	5 AB 2 A 3 BA 2