Problem A. Bandits

Input file: `bandits.in`
Output file: `bandits.out`

After robbing a caravan on a road, m bandits have obtained a loot of n similar diamonds. They decided to divide the loot. To do so they have ordered themselves by their birthdate, and make proposals in turn.

When the proposal is made, the bandits vote either for, or against it. If the strict majority of bandits votes for the proposal, it is accepted, in the other case the author of the proposal is killed.

The proposal states for each of the still alive bandits, what number of diamonds should he get. All bandits are smart, greedy, bloodthirsty and careful. That means, that the bandit votes against the proposal if and only if he is certain that he will survive and get the same number of diamonds or more in future assuming that all the other bandits also vote optimally.

Find out what is the maximal number of diamonds the youngest bandit (the one who makes the first proposal) can get. If he is killed regardless of his proposal, output "-1".

Input

Input file contains m and n ($1 \le m \le 2000$, $1 \le n \le 2000$).

Output

Print the maximal number of diamonds the first bandit can get, or "-1" if he is killed regardless of his proposal.

Example

bandits.in	bandits.out
5 1000	997
2 1000	-1

Problem B. Fitness Club

Input file: `fitness.in`
Output file: `fitness.out`

Fitness clubs are very popular among citizens of the capital of Flatland. In fitness club "Flat" there are k lockers where club visitor can put his belongings for the time of training.

During the day there are n trainings in the fitness club. Each visitor comes to the beginning of some training (at the time he or she gets a key for his locker) and leaves after the end of the training (at the time he or she returns key to the reception). Schedule is organized in such a way that all visitors attending the training leave before the first visitor for the next training comes.

Some of the visitors close the locker before leaving and some of them do not close. All visitors of the club visit the club frequently, so staff of the club know whether each visitor closes the locker. Therefore for each training two numbers are known: number a_i of visitors who will close the locker and number b_i of visitors who will not.

In the beginning of the day all k lockers are closed. Staff of the club want to maximize the nnumber of lockers closed at the end of the day — in this case they will have to do less to prepare the club to the next day. To achieve this goal it may be reasonable to give the key from the opened locker to a person who will definitely close it.

You are to find the maximal number of lockers which can be closed at the end of the day in the case of optimal actions of the staff.

Input

First line of the input file contains two integer numbers: n ($1 \le n \le 100$) and k ($1 \le k \le 1000$). Each of the following n lines contains two integer numbers a_i and b_i ($0 \le a_i, b_i \le k$, $a_i + b_i \le k$).

Output

Output the answer for the problem.

Examples

fitness.in	fitness.out
2 4 1 2 1 1	3

Let us number lockers with integer numbers from 1 to 4. Keys for the first training visitors should be distributed in such a way: locker 1 — to person who will definitely close the locker, lockers 2 and 3 — to those who will definitely not close the locker. Keys for the second training visitors should be distributed in such a way: locker 2 — to person who will definitely close the locker, locker 3 — to the person who will definitely not close the locker. In the end of the day only locker (locker 3) will be closed.

Problem C. Graduated Lexicographical Ordering

Input file: `grlex.in`
Output file: `grlex.out`

Consider integer numbers from 1 to n. Let us call the sum of digits of an integer number its *weight*. Denote the weight of the number x as $w(x)$.

Now let us order the numbers using so called *graduated lexicographical ordering*, or shorter *grlex ordering*. Consider two integer numbers a and b. If $w(a) < w(b)$ then a goes before b in grlex ordering. If $w(a) = w(b)$ then a goes before b in grlex ordering if and only if the decimal representation of a is lexicographically smaller than the decimal representation of b.

Let us consider some examples.

$120 <_{grlex} 4$ since $w(120) = 1 + 2 + 0 = 3 < 4 = w(4)$.

$555 <_{grlex} 78$ since $w(555) = 15 = w(78)$ and "`555`" is lexicographicaly smaller than "`78`".

$20 <_{grlex} 200$ since $w(20) = 2 = w(200)$ and "`20`" is lexicographicaly smaller than "`200`".

Given n and some integer number k, find the position of the number k in grlex ordering of integer numbers from 1 to n, and the k-th number in this ordering.

Input

Input file contains n and k ($1 \le k \le n \le 10^{18}$).

Output

On the first line print the position of the number k in grlex ordering of integer numbers from 1 to n. On the second line print the integer number that occupies the k-th position in this ordering.

Example

grlex.in	grlex.out
20 10	2 14

Problem D. Network Wars

Input file: `network.in`
Output file: `network.out`

Network of Byteland consists of n servers, connected by m optical cables. Each cable connects two servers and can transmit data in both directions. Two servers of the network are especially important — they are connected to global world network and president palace network respectively.

The server connected to the president palace network has number 1, and the server connected to the global world network has number n.

Recently the company *Max Traffic* has decided to take control over some cables so that it could see what data is transmitted by the president palace users. Of course they want to control such set of cables, that it is impossible to download any data from the global network to the president palace without transmitting it over at least one of the cables from the set.

To put its plans into practice the company needs to buy corresponding cables from their current owners. Each cable has some cost. Since the company's main business is not spying, but providing internet connection to home users, its management wants to make the operation a good investment. So it wants to buy such a set of cables, that cables *mean cost* is minimal possible.

That is, if the company buys k cables of the total cost c, it wants to minimize the value of c/k.

Input

The first line of the input file contains n and m ($2 \le n \le 100$, $1 \le m \le 400$). Next m lines describe cables — each cable is described with three integer numbers: servers it connects and the cost of the cable. Cost of each cable is positive and does not exceed 10^7.

Any two servers are connected by at most one cable. No cable connects a server to itself. The network is guaranteed to be connected, it is possible to transmit data from any server to any other one.

Output

First output k — the number of cables to buy. After that output the cables to buy themselves. Cables are numbered starting from one in order they are given in the input file.

Example

network.in	network.out
6 8 1 2 3 1 3 3 2 4 2 2 5 2 3 4 2 3 5 2 5 6 3 4 6 3	4 3 4 5 6
4 5 1 2 2 1 3 2 2 3 1 2 4 2 3 4 2	3 1 2 3

Problem E. N-gons

Input file: `ngon.in`
Output file: `ngon.out`

One of the toy manufacturing firms of Flatland is going to release the new game for children. The set for this game consists of sticks from which children can compose different figures. However on the presentation of this new product one psychologist mentioned that composing non-degenerate n-gons can be harmful for children. So, it was decided that there should not be a subset, from which an n-gon can be constructed, in the game set.

Game sets are manufactured in such a way that lengths of sticks are positive integer numbers not exceeding k. You are to write a program which constructs a game set of maximal possible size.

Input

Input file contains two integer numbers: n — number of vertices in forbidden n-gons and k — maximal length of a stick ($3 \le n \le 10$, $1 \le k \le 10^8$).

Output

First line of the output file should contain one number — maximal possible size of the set of sticks.

On the second line output lengths of these sticks in non-increasing order.

Example

ngon.in	ngon.out
3 7	5 1 1 2 3 5

Problem F. Numbers to Numbers

Input file: `numbers.in`
Output file: `numbers.out`

Numbers in English are written down in the following way (only numbers less then 10^9 are considered). Number $\overline{abc\,def\,ghi}$ is written as "[abc] million [def] thousand [ghi]". Here "[xyz]" means the written down number $\overline{xyz}$.

In the written down number the part "[abc] million" is omitted if $\overline{abc} = 0$, "[def] thousand" is omitted if $\overline{def} = 0$, and "[ghi]" is omitted if $\overline{ghi} = 0$. If the whole number is equal to 0 it is written down as "zero". Note that words "million" and "thousand" are singular even if the number of millions or thousands respectively is greater than one.

Numbers under one thousand are written down in the following way. The number $\overline{xyz}$ is written as "[x] hundred and [yz]. Here "[x] hundred and" is omitted if $x = 0$. Note that "hundred" is also always singular.

Numbers under 20 are written down as "zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", and "nineteen" respectively. Numbers from 20 to 99 are written down in the following way. Number $\overline{xy}$ is written as "[x0] [y]", and numbers divisible by ten are written as "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", and "ninety" respectively.

For example, number 987 654 312 is written down as "nine hundred and eighty seven million six hundred and fifty four thousand three hundred and twelve", number 100 000 037 as "one hundred million thirty seven", number 1 000 as "one thousand". Note that "one" is never omitted for millions, thousands and hundreds.

Given English text with all numbers written as words, transform it to the text where all numbers are written as numbers. For example, the text "he had one hundred and ninety five dogs" must be tranformed to "he had 195 dogs". You must perform a transformation in such a way that as many words as possible are transformed to numbers, so, for example, "three thousand" must be transformed to "3000", not to "3 thousand".

If there are several way to perform the transformation in such a way, you must do it so that the first number is as great as possible, for example "two thousand thirty two" must be transformed to "2032", not "2030 2". If there are still several ways, maximize the second number, and so on.

Input

Input file contains an English text that only contains English letters, punctuation marks (',', '.', '!', '?', ':', ';', '(', ')'), and blanks (spaces and line feeds).

You must perform the transformation in the way described. Words in numbers to be transformed are separated with blanks only (so, for example, "thirty, three" may only be transformed as "30, 3", not as "33").

When performing transformation you must replace characters starting from the first letter of the first word in the number to the last one in the last word with digits.

Input file is not empty and its size does not exceed 20000 bytes.

Output

Output the result of the transformation.

Example

numbers.in

```
From three thousand one hundred and fifty teams selected
from one thousand four hundred and eleven universities
in seventy five countries competing at one hundred and
twenty seven sites and hundreds more competing at
preliminary contests worldwide, seventy three teams
of students competed for bragging rights and prizes
at The Twenty Eighth Annual ACM International Collegiate
Programming Contest World Finals sponsored by IBM on
March Thirty One, Two Thousand Four, and hosted at the
Obecni Dum, Prague by Czech Technical University in Prague.
```

numbers.out

```
From 3150 teams selected
from 1411 universities
in 75 countries competing at 127 sites and hundreds more competing at
preliminary contests worldwide, 73 teams
of students competed for bragging rights and prizes
at The 20 Eighth Annual ACM International Collegiate
Programming Contest World Finals sponsored by IBM on
March 31, 2004, and hosted at the
Obecni Dum, Prague by Czech Technical University in Prague.
```

Problem G. Beautiful Permutation

Input file: `perm.in`
Output file: `perm.out`

Consider a permutation of integer numbers from 1 to n. Let us call length of the longest monotonic subsequence of the permutation its *ugliness*.

For example, the ugliness of the permutation $\langle 1, 2, 5, 3, 4\rangle$ is 4 because it has a monotonic subsequence $(1, 2, 3, 4)$ of length 4, but has none of length 5. The ugliness of the permutation $\langle 5, 6, 3, 4, 1, 2\rangle$ is 3 because it has a monotonic subsequence $(5, 3, 1)$ of length 3.

Let us call *beautiful* those permutations that have a smallest possible ugliness for given n. Given n, you must find the first in lexicographic order beautiful permutation of size n.

Input

Input file contains n ($1 \le n \le 10\,000$).

Output

Output the first in lexicographic order beautiful permutation of size n.

Example

perm.in	perm.out
2	1 2
4	2 1 4 3

Problem H. Beatiful Numbers

Input file: `phones.in`
Output file: `phones.out`

Many companies use "beautiful" phone numbers in advertising, because such numbers are easier to remember. But what can we do if the number does not appear to be beatiful? We can look at the number more carefully and regroup digits so the number becomes more "beatiful". For example, for the phone number 872-73-33 more beautiful regrouping is 8727-333.

Consider the following method of measuring "beauty" of the phone number. The number should be partitioned into several groups of digits each of them containing 2 to 4 digits. "Beauty" of the partition is the sum of points for each group. Points for each type of groups are listed in the table.

Group pattern	Points
aa	2
aba	2
aab, abb	2
aaa	3
abac, baca	2
abab	3
aabb	3
abba	4
baaa, abaa, aaba, aaab	3
aaaa	5

In this table by characters "a", "b" and "c" are denoted different digits. For example, "223" and "667" fit the pattern "aab", but "123" and "888" do not fit.

Given a phone number find the most "beatiful" partition of it.

Input

Input file contains one string consisting of 7 digits — the given phone number.

Output

In the first line of the output file output the most "beatiful" partition of the phone number. In the second line output the value of "beauty" for this partition.

If there are several possible solutions output any of them.

Example

phones.in	phones.out
8727333	8727-333 5
8827291	88-272-91 4

Problem I. Flipping Bits

Input file: `flippingbits.in`
Output file: `flippingbits.out`

Goose Tattarrattat has a sequence b of bits. The length of the sequence is n. Tattarrattat also has a favorite integer m which is between 1 and n, inclusive.

A sequence of n bits is called a rotator sequence if it has the following property: its prefix of length $n-m$ is equal to its suffix of length $n-m$.

For example, let $m = 2$. Consider the sequence $b = 10101010$. Its length is $n = 8$, so we have $n - m = 6$. The prefix of length 6 is "101010", the suffix of length 6 is "101010". They are the same, so this b is a rotator sequence. Now consider $b = 11010100$. For this b we compare the prefix "110101" to the suffix "010100". They differ, so this b is not a rotator sequence.

Tattarrattat wants to change her sequence b into some rotator sequence. She will produce such a sequence in a sequence of steps. In each step she can do one of the following two types of changes to the sequence:

- Flip any bit (from 1 to 0 or from 0 to 1).
- Flip the first $k \cdot m$ bits, for any positive integer k.

You are given b: each of its characters will be either '0' or '1'. Find the minimal number of steps required to obtain some rotator sequence.

Input

The first line contains sequence b ($1 \leq |b| \leq 300$). Sequence b consists of n characters '0' and '1'. The next line contains integer m ($1 \leq m \leq n$).

Output

Output the minimal number of steps required to obtain some rotator sequence.

Examples

flippingbits.in	flippingbits.out
00111000 1	2
101100001101 3	2
11111111 4	0
1101001000 8	1
11011100101110 5	4

Problem J. Puzzle

Input file: `puzzle.in`
Output file: `puzzle.out`

Board games with tokens and large paper maps were quite popular a few years ago.

Recenlty Vasya had found k maps for such games on the attic. Without descriptions he was not able to understand the rules of the games, so he has invented his own rules.

On each of the maps there is a number of circles — possible token positions. One of these positions is marked as "Start" position and some other position — as "Finish". Some of the circles are connected by segments and two positions can be connected by more than one segment. A player can move token from one position to another if and only if they are connected by a segment.

To play the game invented by Vasya one needs all the maps Vasya found. On each map the player puts a token to the "Start" position. On each move the player should move tokens on all maps. Each token should be move to position connected by a segment with the current position of the token. The token should be moved even it is in the "Finish" position.

Vasya wonders what is the minimal number of moves needed to put tokens an all maps to "Finish" positions.

It is guaranteed that on each map every position is reachable from all other positions.

Input

First line of the input file contains an integer number k ($1 \le k \le 10$).

Descriptions of these k maps follow. First string of each description contains two integer numbers n_i and m_i ($2 \le n_i \le 50$, $1 \le m_i \le 1500$) — number of positions and segments of i-th map respectively. Positions are numbered with integer numbers from 1 to n, "Start" position is position number 1, "Finish" position has the number n_i. Each of the next m_i lines contains two integer numbers — numbers of positions connected by a segment.

Output

If there is a sequence a moves leading to the situation in which all tokens are in "Finish" positions, output the minimal length of such a sequence. If such a sequence does not exist, output "`Impossible`".

Example

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

Problem K. Restore the Tree

Input file: `restore.in`
Output file: `restore.out`

An undirected connected graph with no cycles is called a *tree*. A vertex with degree equal to one in a tree is called a *leaf*.

Consider a tree. For each pair of leaves one can determine the distance between these leaves — the number of edges one needs to walk to get from one leaf to another. Given these distances for all pairs of leaves, you need to restore the tree.

Input

The first line of the input file contains l — the number of leaves in the tree ($2 \le l \le 200$). Next l lines contain l integer numbers each — distances between leaves. It is guaranteed that no distance exceeds 200, all distances are non-negative, distance from a leaf to itself is zero, and the distance from leaf i to leaf j is equal to the distance from leaf j to leaf i.

Output

If it is impossible to restore the tree because no such tree exists, output -1 on the first line of the output file.

In the other case first output n — the number of vertices in the tree. After that output $n - 1$ pairs of numbers — edges of the tree, each specified with two vertices it connects. If there are several solutions, output any. First l vertices must correspond to tree leaves as they are described in the input file, other vertices may be numbered in arbitrary order.

Example

restore.in	restore.out
3 0 2 3 2 0 3 3 3 0	5 1 4 2 4 3 5 4 5

Problem L. String

Input file: `string.in`
Output file: `string.out`

So called *recurrent equations* are widely used in mathematics. Usually such equations are used to specify integer sequences. An example of such sequence is the Fibonacci sequence ($F_i = F_{i-1} + F_{i-2}$).

Using the recurrent equations one can define not only integer sequences, but also sequences of strings. In this problem we consider a sequence of strings s_0, s_1, ..., defined in the following way.

String s_0 is an empty string and each string s_i ($i \geq 1$) is obtained from s_{i-1} using the following rule: if decimal representation of i is a substring of s_{i-1}, then $s_i = s_{i-1}$, otherwise s_i is the concatenation of s_{i-1} and decimal representation of i.

You are given the number n. Find the string s_n.

Input

Input file contains an integer number n ($1 \leq n \leq 500$).

Output

Output s_n.

Examples

string.in	string.out
1	1
3	123
13	123456789101113

Problem M. Shooting Game

Input file: `shooting.in`
Output file: `shooting.out`

Alice and Bob play shooting game on a plane.

The game proceeds as follows. There are n obstacles on the plane, each obstacle is a segment. First, Alice selects some point on the plane that doesn't belong to a line containing an obstacle and stands there. Then Bob selects some other point and stands at it. After that Alice shoots at Bob from the gun.

Bob always selects such point that there are as many obstacles as possible between him and Alice. If the line connecting Alice and Bob passes through the end of an obstacle, this obstacle is considered to be between Alice and Bob.

Alice would like to shoot Bob very much, so she tries to select such point that Bob couldn't hide behind too many obstacles. Help Alice to choose such point on the plane, that after Bob selects his point, the number of obstacles between Alice and Bob was minimal possible.

Input

The first line of the input file contains n — the number of obstacles ($1 \le n \le 8$). The following n lines contain four integer numbers x_1, y_1, x_2, y_2 each — the coordinates of the ends of the corresponding obstacle. Obstacles have no common points. Coordinates do not exceed 100 by their absolute values.

Output

The first line of the output file must contain k — the minimal number of obstacles Alice can ensure to be between her and Bob. The second line of the output file must contain two real numbers — coordinates of the point Alice should choose. The point must not belong to any line containing an obstacle. Print as many digits after the decimal point as possible.

Examples

shooting.in	shooting.out
2 0 0 2 0 0 2 2 2	1 1.000000000000000000 1.000000000000000000