Codeforces Round #122 (Div. 2)

A. Exams

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One day the Codeforces round author sat exams. He had n exams and he needed to get an integer from 2 to 5 for each exam. He will have to re-sit each failed exam, i.e. the exam that gets mark 2.

The author would need to spend too much time and effort to make the sum of his marks strictly more than k. That could have spoilt the Codeforces round. On the other hand, if the sum of his marks is strictly less than k, the author's mum won't be pleased at all.

The Codeforces authors are very smart and they always get the mark they choose themselves. Also, the Codeforces authors just hate re-sitting exams.

Help the author and find the minimum number of exams he will have to re-sit if he passes the exams in the way that makes the sum of marks for all n exams equal exactly k.

Input

The single input line contains space-separated integers n and k ($1 \le n \le 50$, $1 \le k \le 250$) — the number of exams and the required sum of marks.

It is guaranteed that there exists a way to pass n exams in the way that makes the sum of marks equal exactly k.

Output

Print the single number — the minimum number of exams that the author will get a 2 for, considering that the sum of marks for all exams must equal k.

Examples

input
4 8
output
4

input
4 10
output
2

input
1 3
output
0

Note

In the first sample the author has to get a 2 for all his exams.

In the second sample he should get a 3 for two exams and a 2 for two more.

In the third sample he should get a 3 for one exam.

B. Square

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a square painted on a piece of paper, the square's side equals n meters. John Doe draws crosses on the square's perimeter. John paints the first cross in the lower left corner of the square. Then John moves along the square's perimeter in the clockwise direction (first upwards, then to the right, then downwards, then to the left and so on). Every time he walks $(n + 1)$ meters, he draws a cross (see picture for clarifications).

John Doe stops only when the lower left corner of the square has two crosses. How many crosses will John draw?

The figure shows the order in which John draws crosses for a square with side 4. The lower left square has two crosses. Overall John paints 17 crosses.

Input

The first line contains integer t ($1 \le t \le 10^4$) — the number of test cases.

The second line contains t space-separated integers n_i ($1 \le n_i \le 10^9$) — the sides of the square for each test sample.

Output

For each test sample print on a single line the answer to it, that is, the number of crosses John will draw as he will move along the square of the corresponding size. Print the answers to the samples in the order in which the samples are given in the input.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
3 4 8 100
output
17 33 401

C. Cutting Figure

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You've gotten an $n \times m$ sheet of squared paper. Some of its squares are painted. Let's mark the set of all painted squares as A. Set A is connected. Your task is to find the minimum number of squares that we can delete from set A to make it not connected.

A set of painted squares is called *connected*, if for every two squares a and b from this set there is a sequence of squares from the set, beginning in a and ending in b, such that in this sequence any square, except for the last one, shares a common side with the square that follows next in the sequence. An empty set and a set consisting of exactly one square are connected by definition.

Input

The first input line contains two space-separated integers n and m ($1 \le n, m \le 50$) — the sizes of the sheet of paper.

Each of the next n lines contains m characters — the description of the sheet of paper: the j-th character of the i-th line equals either "#", if the corresponding square is painted (belongs to set A), or equals "." if the corresponding square is not painted (does not belong to set A). It is guaranteed that the set of all painted squares A is connected and isn't empty.

Output

On the first line print the minimum number of squares that need to be deleted to make set A not connected. If it is impossible, print -1.

Examples

input
5 4 #### #..# #..# #..# ####
output
2

input
5 5 ##### #...# ##### #...# #####
output
2

Note

In the first sample you can delete any two squares that do not share a side. After that the set of painted squares is not connected anymore.

The note to the second sample is shown on the figure below. To the left there is a picture of the initial set of squares. To the right there is a set with deleted squares. The deleted squares are marked with crosses.

D. Xor

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

John Doe has four arrays: a, b, k, and p. Each array consists of n integers. Elements of all arrays are indexed starting from 1. Array p is a permutation of integers 1 to n.

John invented a game for his friends and himself. Initially a player is given array a. The player must consecutively execute exactly u operations on a. You are permitted to execute the following operations:

- Operation 1: For each $i \in \{1, 2, \ldots, n\}$ change a_i into $a_i \oplus b_i$. Expression $x \oplus y$ means applying the operation of a bitwise xor to numbers x and y. The given operation exists in all modern programming languages, for example, in language *C++* and *Java* it is marked as "^", in *Pascal* — as "xor".
- Operation 2: For each $i \in \{1, 2, \ldots, n\}$ change a_i into $a_{p_i} + r$. When this operation is executed, all changes are made at the same time.

After all u operations are applied, the number of points the player gets is determined by the formula $s = \sum_{i=1}^{i \le n} a_i k_i$.

John wants to find out what maximum number of points a player can win in his game. Help him.

Input

The first line contains space-separated integers n, u and r ($1 \le n, u \le 30$, $0 \le r \le 100$) — the number of elements in each array, the number of operations and the number that describes one of the operations.

Each of the next four lines contains n space-separated integers — arrays a, b, k, p. The first line has array a, the second line has array b, the third line has array k and the fourth one has array p.

It is guaranteed that elements of arrays a and b are positive and do not exceed 10^4 ($1 \le a_i, b_i \le 10^4$), elements of array k do not exceed 10^4 in the absolute value ($|k| \le 10^4$) and p is a permutation of numbers from 1 to n.

Output

On a single line print number s — the maximum number of points that a player can win in John's game.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
3 2 1 7 7 7 8 8 8 1 2 3 1 3 2

output
96

input
2 1 0 1 1 1 1 1 -1 1 2

output
0

Note

In the first sample you should first apply the operation of the first type, then the operation of the second type.

E. Hamming Distance

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Hamming distance between strings a and b of equal length (denoted by $h(a, b)$) is equal to the number of distinct integers i $(1 \le i \le |a|)$, such that $a_i \ne b_i$, where a_i is the i-th symbol of string a, b_i is the i-th symbol of string b. For example, the Hamming distance between strings "aba" and "bba" equals 1, they have different first symbols. For strings "bbba" and "aaab" the Hamming distance equals 4.

John Doe had a paper on which four strings of equal length s_1, s_2, s_3 and s_4 were written. Each string s_i consisted only of lowercase letters "a" and "b". John found the Hamming distances between all pairs of strings he had. Then he lost the paper with the strings but he didn't lose the Hamming distances between all pairs.

Help John restore the strings; find some four strings s'_1, s'_2, s'_3, s'_4 of equal length that consist only of lowercase letters "a" and "b", such that the pairwise Hamming distances between them are the same as between John's strings. More formally, set s'_i must satisfy the condition $\forall i, j \in \{1, 2, 3, 4\}, h(s_i, s_j) = h(s'_i, s'_j)$.

To make the strings easier to put down on a piece of paper, you should choose among all suitable sets of strings the one that has strings of **minimum length**.

Input

The first line contains space-separated integers $h(s_1, s_2)$, $h(s_1, s_3)$, $h(s_1, s_4)$. The second line contains space-separated integers $h(s_2, s_3)$ and $h(s_2, s_4)$. The third line contains the single integer $h(s_3, s_4)$.

All given integers $h(s_i, s_j)$ are non-negative and do not exceed 10^5. It is guaranteed that at least one number $h(s_i, s_j)$ is positive.

Output

Print -1 if there's no suitable set of strings.

Otherwise print on the first line number len — the length of each string. On the i-th of the next four lines print string s'_i. If there are multiple sets with the minimum length of the strings, print any of them.

Examples

input
4 4 4 4 4 4
output
6 aaaabb aabbaa bbaaaa bbbbbb