Problem A. Character Encoding

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

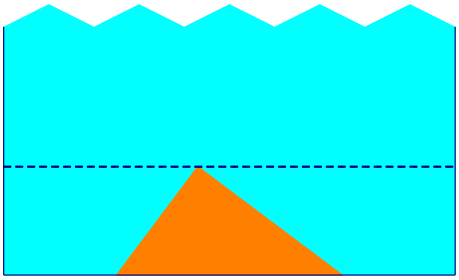
Problem B. Pizza Hub

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	128 megabytes

Coffee Chicken has started a new restaurant named Pizza Hu...b! It provides various styles of pizzas, hamburgers, sandwiches, coffee, chickens and many other awesome Western cuisines. Welcome to Pizza Hub after this Multi-University Training Contest!

Since the pizzas are so exquisite, it is never a bad thing to design nice paper pads for them. The pizzas provided in Pizza Hub are sliced into triangles. The rectangular-shaped paper pads are cut from a paper strip of fixed width which is long enough. The pizza should be placed entirely on the pad; however, their borders are allowed to touch. Also, you are allowed to rotate the pizza.

As the customized paper strip is rather expensive, minimizing the size of the pizza pad can save a lot of money. Can you determine the minimum possible height of the pizza pad, given the width of the paper strip? The following picture illustrates the first sample test case.



Input

The first line of the input is a single integer T ($1 \le T \le 50000$), the number of test cases.

Each test case is a single line of seven integers $x_1, y_1, x_2, y_2, x_3, y_3$ ($0 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 10000$) and w ($1 \le w \le 10000$), where (x_1, y_1), (x_2, y_2) and (x_3, y_3) are Cartesian coordinates of the vertices of the pizza, and w is the width of the strip. It is guaranteed that the three vertices are not collinear.

Output

For each test case, display the minimum height of the pizza pad with an absolute or relative error of no more than 10^{-6}. If it is impossible to make a pizza pad, display `impossible` instead.

Example

standard input	standard output
2 0 0 3 0 0 4 10 0 0 3 0 0 4 1	2.400000000 impossible

Problem C. City Development

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	128 megabytes

Sociologists have proposed a model to predict the development of cities in a country. The model works as follows.

Suppose, that there are n cities in the country numbered 1 through n, and there are k different levels of administrative divisions in this country. City is the lowest level (i.e., the k-th level) administrative division. Every i-th level administrative division has exactly n_i cities, where $n \bmod n_1 = n_1 \bmod n_2 = \cdots = n_{k-1} \bmod n_k = 0$, and $n_k = 1$. Also, two cities numbered a and b belong to the same i-th level administrative division if and only if $\lceil a/n_i \rceil = \lceil b/n_i \rceil$. It is clear that two cities belonging to the same lower level administrative division must belong to the same higher level administrative division.

The model owes the development of a city both to the city itself, and to the mutual interactions between cities. Obviously, the interaction between closer cities is stronger. We introduce the concept of *lowest common administrative level* (LCA for short) to characterize the closeness between two cities. Formally, for two cities numbered a, b, $\mathrm{LCA}(a, b)$ is defined as

$$\mathrm{LCA}(a, b) = \max\{i : a, b \text{ belong to the same } i\text{-th level administrative division}\}$$

In case that city a and b don't belong to any common administrative division, we define $\mathrm{LCA}(a, b) = 0$. Also, we have $\mathrm{LCA}(a, a) = k$. Let $d_t(x)$ denote the development index of the x-th city in the t-th year, then the model says that

$$d_{t+1}(x) = \sum_{i=1}^{n} \rho_{\mathrm{LCA}(x,i)} d_t(i)$$

where ρ_i $(0 \le i \le k)$ is called the interaction coefficient between two cities a, b with $\mathrm{LCA}(a, b) = i$.

Now, given the initial development indexes of all cities, can you use this model to predict the development indexes after T years?

Input

The first line of input consists of only a single integer K $(1 \le K \le 20)$, the number of test cases.

Each test case begins with three integers n, k, T $(1 \le k \le n \le 3 \times 10^5, 1 \le T \le 10^{18})$, the number of cities, the number of levels of administrative regions, and the number of years, respectively. Then follows a line with k integers $n_1, n_2, \cdots, n_k$ $(1 = n_k \le n_{k-1} \le \cdots \le n_2 \le n_1 \le n, n_{i-1} \bmod n_i = n \bmod n_1 = 0)$, the number of cities in each level of administrative divisions. The third line contains n integers $d_0(1), d_0(2), \cdots, d_0(n)$ $(1 \le d_0(i) \le 10^6)$, the initial development indexes of the cities. The last line of each test case contains $k + 1$ integers, $\rho_0, \rho_1, \cdots \rho_k$ $(1 \le \rho_0 \le \rho_1 \le \cdots \le \rho_k \le 10^6)$, the interaction coefficients.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, display a line of n integers, denoting the predicted development indexes of all cities after T years in order. Since the answers might be rather big, you should display these numbers modulo 998244353.

Example

standard input	standard output
1 4 2 1 2 1 1 3 5 6 2 4 5	39 41 57 58

Problem D. Parentheses Matrix

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	128 megabytes

A *parentheses matrix* is a matrix where every element is either '(' or ')'. We define the *goodness* of a parentheses matrix as the number of *balanced* rows (from left to right) and columns (from up to down). Note that:

- an empty sequence is balanced;
- if A is balanced, then (A) is also balanced;
- if A and B are balanced, then AB is also balanced.

For example, the following parentheses matrix is a 2×4 matrix with goodness 3, because the second row, the second column and the fourth column are balanced:

```
)()(
()()
```

Now, give you the width and the height of the matrix, please construct a parentheses matrix with maximum goodness.

Input

The first line of input is a single integer T ($1 \leq T \leq 50$), the number of test cases.

Each test case is a single line of two integers h, w ($1 \leq h, w \leq 200$), the height and the width of the matrix, respectively.

Output

For each test case, display h lines, denoting the parentheses matrix you construct. Each line should contain exactly w characters, and each character should be either '(' or ')'. If multiple solutions exist, you may print any of them.

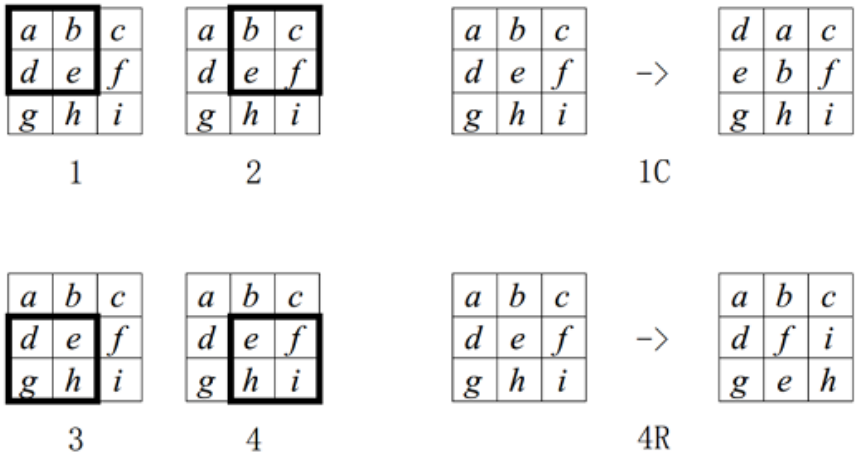
Example

standard input	standard output
3 1 1 2 2 2 3	(())(((( )))

Problem E. Magic Square

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

A magic square is a 3×3 square, where each element is a single digit between 1 and 9 inclusive, and each digit appears exactly once. There are 4 different contiguous 2×2 subsquares in a magic squares, which are labeled from 1 to 4 as the following figure shows. These 2×2 subsquares can be rotated. We use the label of the subsquare with an uppercase letter to represent a rotation. If we rotate the subsquare clockwise, the letter is `'C'`; if we rotate it counterclockwise, the letter is `'R'`. The following figure shows two different rotations.



Now, given the initial state of a magic square and a sequence of rotations, please print the final state of the magic square after these rotations are performed.

Input

The first line of input is a single integer T ($1 \le T \le 100$), the number of test cases.

Each test case begins with a single integer n ($1 \le n \le 100$), the number of rotations. It is then followed by a 3×3 square, where every digit between 1 and 9 inclusive appears exactly once, representing the initial state of the magic square. The following n lines describe the sequence of rotations.

The test data guarantees that the input is valid.

Output

For each test case, display a 3×3 square, denoting the final state of the magic square.

Example

standard input	standard output
1 2 123 456 789 1C 4R	413 569 728

Problem F. Boolean 3-Array

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

In this problem, we are going to deal with a special structure called Boolean 3-array.

A *Boolean 3-array* of size $m \times n \times p$ is a three-dimensional array denoted as A, where $A[i][j][k] \in \{0, 1\}$ $(1 \le i \le m, 1 \le j \le n, 1 \le k \le p)$. We define any one of these as an *operation* on a Boolean 3-array A of size $m \times n \times p$:

- Choose some fixed a $(1 \le a \le m)$, then flip $A[a][j][k]$ for all $1 \le j \le n$, $1 \le k \le p$;
- Choose some fixed b $(1 \le b \le n)$, then flip $A[i][b][k]$ for all $1 \le i \le m$, $1 \le k \le p$;
- Choose some fixed c $(1 \le c \le p)$, then flip $A[i][j][c]$ for all $1 \le i \le m$, $1 \le j \le n$;
- Choose some fixed a_1, a_2 $(1 \le a_1, a_2 \le m)$, then swap $A[a_1][j][k]$ and $A[a_2][j][k]$ for all $1 \le j \le n$, $1 \le k \le p$;
- Choose some fixed b_1, b_2 $(1 \le b_1, b_2 \le n)$, then swap $A[i][b_1][k]$ and $A[i][b_2][k]$ for all $1 \le i \le m$, $1 \le k \le p$;
- Choose some fixed c_1, c_2 $(1 \le c_1, c_2 \le p)$, then swap $A[i][j][c_1]$ and $A[i][j][c_2]$ for all $1 \le i \le m$, $1 \le j \le n$.

Here "filp" means change the value of the element, i.e., replace 0 with 1 and replace 1 with 0.

We say two Boolean 3-arrays A, B are *essentially identical*, if and only if any one of them can be transformed to the other, by applying operations finitely many times; otherwise, we say A and B are *essentially different*.

Now, given the size of the Boolean 3-array, can you determine the maximum number of Boolean 3-arrays of given size you may choose, such that any two of them are essentially different?

Input

The first line of input is a single integer T $(1 \le T \le 300)$, the number of test cases.

Each test case is a single line of three integers n, m, p $(1 \le m, n, p \le 13)$, the size of the Boolean 3-array.

Output

For each test case, display an integer in a single line: the answer modulo 998244353.

Example

standard input	standard output
3 1 1 1 2 2 2 2 3 4	1 9 723

Problem G. Card Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	128 megabytes

Alice and Bob are playing a card game. In this card game, a number is written on each face of the playing card. The rule of the game is described as follows:

- Alice arranges the cards in a row, and for each of the cards, she chooses one of its faces to place it up;
- Bob turns over minimum number of cards to make all numbers on the front faces unique.

They play the game some times, and Bob always succeeds making the numbers unique. However, both of them are not sure whether the number of cards flipped is minimum. Moreover, they want to figure out the number of different ways of turning over minimum number of cards to make the numbers unique. Two ways are considered equal if and only if the sets of flipped cards are equal. Please write a program to help them!

Input

The first line of the input is a single integer T ($1 \leq T \leq 50$), the number of test cases.

Each test case begins with a single line of integer n ($1 \leq n \leq 10^5$), the number of cards on the table. Then follow n lines, specifying the cards that Alice arranges. Each of these n lines contains two integers x, y ($1 \leq x, y \leq 2n$), meaning that Alice places a card with number x on the front face and number y on the back face.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, display two integers in a line, the minimum number of cards to turn over, and the number of different ways of flipping modulo 998244353, respectively. If it is impossible to turn over cards to make all numbers unique, display `-1 -1` instead.

Example

standard input	standard output
3 4 1 2 1 3 4 5 4 6 2 1 1 1 1 3 1 2 3 4 5 6	2 4 -1 -1 0 1

Note

In the first sample test case, Bob may turn over one of the first two cards and one of the last two cards,

so there are four different ways of turning over two cards to make all numbers on the front faces unique. Obviously, this can't be done with less than two cards flipped.

In the second sample test case, it is impossible to make all numbers on the front faces unique.

In the third sample test case, all numbers on the front faces are already distinct.

Problem H. K-Similar Strings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

Acesrc loves solving string problems. He defined a relation called *k-similarity* between two **nonempty** strings.

The definition of k-similarity is shown below:

1. for nonempty string S, S and S are k-similar;
2. for two nonempty strings S and T with $|S| + |T| \leq k$, if $S \circ T$ and $T \circ S$ are k-similar ($\circ$ denotes string concatenation), then S and T are k-similar;
3. if S and T are k-similar, then $P \circ S \circ Q$ and $P \circ T \circ Q$ are k-similar, where P and Q are arbitrary (possibly empty) strings;
4. if S and U are k-similar, U and T are k-similar, then S and T are k-similar.

For example, `aaa` and `aaa` are 3-similar according to the the first condition. Hence, `a` and `aa` are 3-similar according to the second condition. Moreover, `ba` and `baa`, `baa` and `baaa` are also 3-similar, respectively, according to the third condition. Finally, `ba` and `baaa` are 3-similar, according to the fourth condition.

Now, given two strings A, B and an integer k, please determine whether A and B are k-similar.

Input

The first line of the input is a single integer T ($1 \leq T \leq 5000$), denoting the number of test cases. Each test case contains three lines, which represent k ($1 \leq k \leq 10^6$), A, B, respectively. It is guaranteed that A and B contain only lowercase letters `'a'-'z'` and the lengths of A and B are between 1 and 2×10^5, inclusive. It is further guaranteed that the sum of lengths of A and B in all test cases does not exceed 3×10^6.

Output

For each test case, display a single line: `yes` if A and B are k-similar, or `no` if they are not.

Example

standard input	standard output
4 3 ba baa 2 aab ab 1 acesrc acesrc 100 roundgod zyb	yes no yes no

Problem I. Make ZYB Happy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

It's known to all that ZYB is godlike, so obviously he has a large number of titles, such as `jsking`, `bijingzyb` and `nbazyb`. ZYB likes his titles very much.

Each of ZYB's titles is a string consisting of lower case letters `'a'-'z'` associated with a happiness value h_i, which shows how much ZYB likes this title. If you say any substring of some title with happiness value x, he will get x happiness points. Moreover, a string may appear in more than one title. In this case, the happiness points ZYB gets are multiplied. If the string you say is not the substring of any of his titles, he gets no happiness point.

For example, let's say ZYB has two titles: `zybnb` (with happiness value 3) and `ybyb` (with happiness value 5). If you say `y`, `b` or `yb`, ZYB will get 15 happiness points; if you say `z`, `zy` or `zyb`, ZYB will only get 3 happiness points; if you say `ybz` or `ybac` he will get 0 happiness points.

One day, you find ZYB pretty sad. As a big fan of ZYB, you want to say a word to ZYB to cheer him up. However, ZYB is really busy, so you can only say no more than m letters. As you haven't seen ZYB for a long time, you are so excited that you forget what you want to say, so you decide to choose to say a nonempty string no longer than m and only containing `'a'-'z'` with equal probability. You want to know the expectations of happiness points you will bring to ZYB for different m.

Input

The first line contains an integer n ($1 \le n \le 10000$), the number of titles ZYB has.

The i-th of the next n lines contains a nonempty string t_i, which only contains lower case letters `'a'-'z'`, representing the i-th title. The sum of lengths of all titles does not exceed 3×10^5.

Then follows a line with n integers h_i ($1 \le h_i \le 10^6$), the happiness value of i-th title.

The next line is a single integer Q ($1 \le Q \le 3 \times 10^5$), the number of queries.

For the next Q lines, each contains a single integer m ($1 \le m \le 10^6$), meaning that you can say no more than m letters to ZYB.

The input data contains only one test case.

Output

For each query, display a single line of integer, representing the answer. It can be proved that the answer can be uniquely written as p/q where p and q are non-negative integers with $\gcd(p, q) = \gcd(q, 10^9+7) = 1$, and you should display $p \cdot q^{-1} \bmod (10^9 + 7)$, where q^{-1} means the multiplicative inverse of q modulo $10^9 + 7$.

Example

standard input	standard output
2 zybnb ybyb 3 5 4 1 2 3 4	769230776 425925929 891125950 633120399

Note

For the first query, you can bring him 3 happiness points if you say `z` or `n`, and 15 happiness points if you say `y` or `b`; all other strings of length 1 bring no happiness point to ZYB. Therefore, the expectation is $(2 \times 3 + 2 \times 15)/26 = 18/13$, and the answer is $18 \times 13^{-1} \bmod (10^9 + 7) = 769230776$.

Problem J. Taotao Picks Apples

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

There is an apple tree in front of Taotao's house. When autumn comes, n apples on the tree ripen, and Taotao will go to pick these apples.

When Taotao picks apples, Taotao scans these apples from the first one to the last one. If the current apple is the first apple, or it is strictly higher than the previously picked one, then Taotao will pick this apple; otherwise, he will not pick.

Given the heights of these apples $h_1, h_2, \cdots, h_n$, you are required to answer some independent queries. Each query is two integers p, q, which asks the number of apples Taotao would pick, if the height of the p-th apple were q (instead of h_p). Can you answer all these queries?

Input

The first line of input is a single line of integer T ($1 \leq T \leq 10$), the number of test cases.

Each test case begins with a line of two integers n, m ($1 \leq n, m \leq 10^5$), denoting the number of apples and the number of queries. It is then followed by a single line of n integers $h_1, h_2, \cdots, h_n$ ($1 \leq h_i \leq 10^9$), denoting the heights of the apples. The next m lines give the queries. Each of these m lines contains two integers p ($1 \leq p \leq n$) and q ($1 \leq q \leq 10^9$), as described in the problem statement.

Output

For each query, display the answer in a single line.

Example

standard input	standard output
1 5 3 1 2 3 4 4 1 5 5 5 2 3	1 5 3

Note

For the first query, the heights of the apples were $5, 2, 3, 4, 4$, so Taotao would only pick the first apple.

For the second query, the heights of the apples were $1, 2, 3, 4, 5$, so Taotao would pick all these five apples.

For the third query, the heights of the apples were $1, 3, 3, 4, 4$, so Taotao would pick the first, the second and the fourth apples.

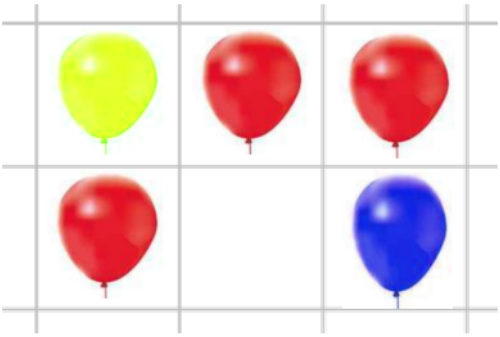
Problem K. Pop the Balloons

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	256 megabytes

Recently, an interesting balloon-popping game can be commonly found near the streets. The rule of this game is quite simple: some balloons are tied to cells of an $n \times m$ lattice, and you are allowed to throw k darts to prick the balloons. The more balloons you pierce, the more incredible prize you will get.

Probably because too many people have got bored with this childish game, a new variation of the game has appeared. In this variation, the balloons are replaced with more powerful ones: when a balloon explodes, strong gusts travel in four directions, blowing away all remaining balloons in the same row and column. In order to reduce the difficulty, not all cells are filled with a balloon.

For example, if you prick the yellow balloon in the following figure, then the red balloons will be blown away, with only the blue balloon remaining.



Now, you are given k darts. Since you are a good dart player that you can precisely throw it to any position you want, it is easy for you to use these k darts to clear all balloons (either by directly pricking, or by blowing away by other exploded balloons). Now you begin to consider: for every $1 \leq x \leq k$, how many different ways are there to clear all balloons with exactly x darts? Two ways are considered different if and only if there exists i, such that the positions of i-th pricked balloons differ. Note that you cannot throw the dart to an empty cell.

Input

The first line of input is a single integer T ($1 \leq T \leq 100$), denoting the number of test cases.

Each test case begins with a line of three integers m, n ($1 \leq m \leq 12, 1 \leq n \leq 20$), denoting the size of the balloon lattice, and k ($1 \leq k \leq 20$), denoting the number of darts. The next m lines, each containing n characters, describe the balloon lattice. Each character is either '`Q`', which denotes a balloon, or '`.`', which denotes an empty cell.

Output

For each test case, display k lines. The i-th line is a single integer denoting the number of different ways of clearing all balloons with exactly i darts.

Example

standard input	standard output
4 2 2 2 QQ .Q 2 2 2 QQ .. 3 3 3 .Q. QQQ .Q. 1 3 1 Q.Q	1 2 2 0 1 8 0 2

Problem L. From ICPC to ACM

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

You, a former International Collegiate Programming Contest player, have now become an Assistant Cost Manager of Advanced Computer Manufacturing company. There are so many different kinds of costs you should deal with, including material cost, manufacturing cost and warehouse cost. Also, the customer's demand must be fully satisfied. Moreover, the space of warehouse is limited. To make things worse, all these parameters vary with time. To resolve this matter once for all, you decide to use your algorithmic knowledge to write a program.

Specifically, you have to find an optimal operational planning for the next k months, under the following constraints: **in the i-th month**,

- the price of raw materials is c_i yuan per unit, and the quantity of raw materials you purchase is not limited.
- the customer's demand is d_i computers, that is, you must sell exactly d_i computers to customers this month.
- your company can manufacture at most p_i computers. Manufacturing one computer consumes one unit of raw materials plus a manufacturing cost of m_i yuan.
- you can store raw materials and computers in the warehouse for future use. According to the regulations of your company, the raw materials and computers must be placed in two different areas of the warehouse. You can store at most e_i computers to the next month. However, since the volume of raw materials is negligible, the number of raw materials you store is not limited. Storing one unit of raw materials or one computer to the next month takes R_i yuan or E_i yuan, respectively.

Also,

- you may immediately use the raw materials you purchase this month to manufacture computers, as long as the the production capacity of this month is enough; likewise, you may manufacture and sell computers in the same month. These raw materials and computers do not take up warehouse space.
- initially, there are no raw materials or computers in your company.

Your program should report whether all customer's demands can be fully satisfied, and, if so, report the minimum total cost.

Input

The first line of input is a single integer T ($1 \le T \le 200$), denoting the number of test cases.

Each test case begins with a single line of integer k ($2 \le k \le 50000$), the number of months. The i-th of the following k lines contains four integers c_i, d_i, m_i, p_i ($0 \le c_i, d_i, m_i, p_i \le 10^4$), the price of raw materials, the customer's demand, the unit manufacturing cost and the manufacturing capacity of the i-th month, respectively. The i-th of the next $k-1$ lines contains three integers e_i, R_i, E_i ($0 \le e_i \le 10^8, 0 \le R_i, E_i \le 10^4$), the capacities of computer area of the warehouse, and the warehouse cost of storing a unit of raw material and a computer, between the i-th and the $(i+1)$-th month, respectively.

It is guaranteed that the sum of k over all test cases does not exceed 3×10^5.

Output

For each test case, display an integer in a line, denoting the minimum total cost. If customer demands cannot be fully satisfied, display `-1` instead.

Example

standard input	standard output
2 2 10 5 3 6 15 7 2 8 2 3 2 2 0 8 0 7 0 0 0 0 0 0 0	170 -1

Note

In the first sample test case, you may purchase 12 units of raw materials, which takes 120 yuan, and put 7 of them into the warehouse for the next month, which takes 21 yuan. Then, your company may manufacture 5 computers and sell them, which takes 15 yuan. In the second month, your company may manufacture and sell 7 computers using the raw materials in the warehouse, which takes 14 yuan. The total cost is 170 yuan, which can be proved to be optimal.

In the second sample test case, the manufacturing capacity of the first month is less than the customer's demand, so the customer's demand cannot be fully satisfied.