A. Countdown

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The "Countdown" TV show has a part that consists of obtaining a number by combining six different numbers using the basic mathematical operations: addition, subtraction, product, and division. The basic rules for the game are:

- The contestant selects six of twenty-four shuffled tiles. The tiles are arranged into two groups: four "large numbers" (25, 50, 75 and 100) and the remainder "small numbers", which comprise two each of the numbers 1 to 10. Hence the tiles have the values {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100}.
- The contestant chooses how many large numbers are in the selection; anywhere from none.
- The contestants then have thirty seconds to get a number as close to the target as possible by combining the six selected numbers using addition, subtraction, multiplication, and division.
- Not all numbers need to be used.
- A number can be used as many times as it appears.
- Fractions are not allowed, only positive integers may be used at any stage of the calculation.

Example

- Contestant requests two large numbers and four small numbers.
- Selection is: $75\ 50\ 2\ 3\ 8\ 7$
- Randomly generated target is: 812
- Contestant declares result: 813
- Contestant gives details: $75 + 50 = 125$; $125 - 8 = 117$; $117 \times 7 = 819$; $3 \times 2 = 6$; $819 - 6 = 813$
- Expert notes: $50 + 8 = 58$; $7 \times 2 = 14$; $14 \times 58 = 812$

Your task is to write a program that calculates the best sequence of operations that lead to the target number T. If there is no way to get T, give the closest solution.

Input

The input consists of several cases, one per line. The first line indicates the number of cases C to process ($1 \leq C \leq 50$). Each of the following C lines contains six natural numbers from the set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100} and another natural number T ($1 \leq T \leq 999$) that indicates the target.

Output

The output for each case will be a set of lines with the following format:

- First line: **Target**: number T
- n lines: sequence of operations, the format is $\texttt{operand}_2\ \texttt{operator}\ \texttt{operand}_2 = \texttt{result}$
- Last line: **Best approx**: number obtained
- Blank line

See example for a better understanding. If there is more than one best approximation, all of them will be considered valid. The sequence of operations should be valid, you should never use a value before you obtain it. It is OK to print more operations than needed as long as they are valid. Note that all the numbers and operators must be separated by at least one space

Example

input

```
3
1 75 100 5 3 25 25
100 100 100 100 100 75 345
1 3 1 10 100 75 345
```

output

```
Target: 25
Best approx: 25

Target: 345
100 + 100 = 200
100 + 75 = 175
200 * 175 = 35000
35000 - 100 = 34900
34900 / 100 = 349
Best approx: 349

Target: 345
1 + 3 = 4
100 - 10 = 90
4 - 1 = 3
90 * 3 = 270
75 + 270 = 345
Best approx: 345
```

B. Restoring Three Numbers

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has guessed three positive integers a, b and c. He keeps these numbers in secret, but he writes down four numbers on a board in arbitrary order — their pairwise sums (three numbers) and sum of all three numbers (one number). So, there are four numbers on a board in random order: $a+b$, $a+c$, $b+c$ and $a+b+c$.

You have to guess three numbers a, b and c using given numbers. Print three guessed integers in any order.

Pay attention that some given numbers a, b and c can be equal (it is also possible that $a=b=c$).

Input

The only line of the input contains four positive integers x_1, x_2, x_3, x_4 ($2 \le x_i \le 10^9$) — numbers written on a board in random order. It is guaranteed that the answer exists for the given number x_1, x_2, x_3, x_4.

Output

Print such positive integers a, b and c that four numbers written on a board are values $a+b$, $a+c$, $b+c$ and $a+b+c$ written in some order. Print a, b and c in any order. If there are several answers, you can print any. It is guaranteed that the answer exists.

Examples

input

```
3 6 5 4
```

output

```
2 1 3
```

input

```
40 40 40 60
```

output

```
20 20 20
```

input

```
201 101 101 200
```

output

```
1 100 100
```

C. Draw Brackets!

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A sequence of square brackets is regular if by inserting symbols "`+`" and "`1`" into it, you can get a regular mathematical expression from it. For example, sequences "`[[]][]`", "`[]`" and "`[[][[]]]`" — are regular, at the same time "`][`", "`[[]`" and "`[[]]][`" — are irregular.

Draw the given sequence using a minimalistic pseudographics in the strip of the lowest possible height — use symbols '`+`', '`-`' and '`|`'. For example, the sequence "`[[][]][]`" should be represented as:

```
+-        -++- -+
|+- -++- -+||   |
||   ||   |||   |
|+- -++- -+||   |
+-        -++- -+
```

Each bracket should be represented with the hepl of one or more symbols '`|`' (the vertical part) and symbols '`+`' and '`-`' as on the example which is given above.

Brackets should be drawn without spaces one by one, only dividing pairs of consecutive pairwise brackets with a single-space bar (so that the two brackets do not visually merge into one symbol). The image should have the minimum possible height.

The enclosed bracket is always smaller than the surrounding bracket, but each bracket separately strives to maximize the height of the image. So the pair of final brackets in the example above occupies the entire height of the image.

Study carefully the examples below, they adequately explain the condition of the problem. Pay attention that in this problem the answer (the image) is unique.

Input

The first line contains an even integer n ($2 \le n \le 100$) — the length of the sequence of brackets.

The second line contains the sequence of brackets — these are n symbols "`[`" and "`]`". It is guaranteed that the given sequence of brackets is regular.

Output

Print the drawn bracket sequence in the format which is given in the condition. Don't print extra (unnecessary) spaces.

Examples

input

```
8
[[][]][]
```

output

```
+-        -++- -+
|+- -++- -+||   |
||   ||   |||   |
|+- -++- -+||   |
+-        -++- -+
```

input

```
6
[[[]]]
```

output

```
+-     -+
|+-   -+|
||+- -+||
|||   |||
||+- -+||
|+-   -+|
+-     -+
```

input

```
6
[[][]]
```

output

```
+-        -+
|+- -++- -+|
||   ||   ||
|+- -++- -+|
+-        -+
```

input

```
2
[]
```

output

```
+- -+
|   |
+- -+
```

input

```
4
[][]
```

output

```
+- -++- -+
|   ||   |
+- -++- -+
```

D. Pashmak and Graph

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Pashmak's homework is a problem about graphs. Although he always tries to do his homework completely, he can't solve this problem. As you know, he's really weak at graph theory; so try to help him in solving the problem.

You are given a weighted directed graph with n vertices and m edges. You need to find a path (perhaps, non-simple) with maximum number of edges, such that the weights of the edges increase along the path. In other words, each edge of the path must have strictly greater weight than the previous edge in the path.

Help Pashmak, print the number of edges in the required path.

Input

The first line contains two integers n, m $(2 \le n \le 3 \cdot 10^5; 1 \le m \le min(n \cdot (n - 1), 3 \cdot 10^5))$. Then, m lines follows. The i-th line contains three space separated integers: u_i, v_i, w_i $(1 \le u_i, v_i \le n; 1 \le w_i \le 10^5)$ which indicates that there's a directed edge with weight w_i from vertex u_i to vertex v_i.

It's guaranteed that the graph doesn't contain self-loops and multiple edges.

Output

Print a single integer — the answer to the problem.

Examples

input

```
3 3
1 2 1
2 3 1
3 1 1
```

output

```
1
```

input

```
3 3
1 2 1
2 3 2
3 1 3
```

output

```
3
```

input

```
6 7
1 2 1
3 2 5
2 4 2
2 5 2
2 6 9
5 4 3
4 3 4
```

output

```
6
```

Note

In the first sample the maximum trail can be any of this trails: $1 \to 2, 2 \to 3, 3 \to 1$.

In the second sample the maximum trail is $1 \to 2 \to 3 \to 1$.

In the third sample the maximum trail is $1 \to 2 \to 5 \to 4 \to 3 \to 2 \to 6$.

E. Checkout Assistant

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bob came to a cash & carry store, put n items into his trolley, and went to the checkout counter to pay. Each item is described by its price c_i and time t_i in seconds that a checkout assistant spends on this item. While the checkout assistant is occupied with some item, Bob can steal some other items from his trolley. To steal one item Bob needs exactly 1 second. What is the minimum amount of money that Bob will have to pay to the checkout assistant? Remember, please, that it is Bob, who determines the order of items for the checkout assistant.

Input

The first input line contains number n $(1 \le n \le 2000)$. In each of the following n lines each item is described by a pair of numbers t_i, c_i $(0 \le t_i \le 2000,\ 1 \le c_i \le 10^9)$. If t_i is 0, Bob won't be able to steal anything, while the checkout assistant is occupied with item i.

Output

Output one number — answer to the problem: what is the minimum amount of money that Bob will have to pay.

Examples

input

```
4
2 10
0 20
1 5
1 3
```

output

```
8
```

input

```
3
0 1
0 10
0 100
```

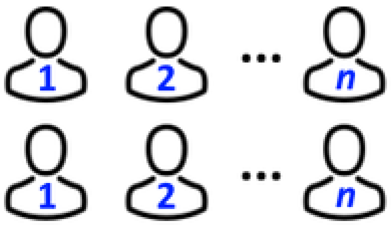
output

```
111
```

F. Basketball Exercise

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

Finally, a basketball court has been opened in SIS, so Demid has decided to hold a basketball exercise session. $2 \cdot n$ students have come to Demid's exercise session, and he lined up them into two rows of the same size (there are exactly n people in each row). Students are numbered from 1 to n in each row in order from left to right.



Now Demid wants to choose a team to play basketball. He will choose players from left to right, and the index of each chosen player (excluding the first one **taken**) will be strictly greater than the index of the previously chosen player. To avoid giving preference to one of the rows, Demid chooses students in such a way that no consecutive chosen students belong to the same row. The first student can be chosen among all $2n$ students (there are no additional constraints), and a team can consist of any number of students.

Demid thinks, that in order to compose a perfect team, he should choose students in such a way, that the total height of all chosen students is maximum possible. Help Demid to find the maximum possible total height of players in a team he can choose.

Input

The first line of the input contains a single integer n ($1 \le n \le 10^5$) — the number of students in each row.

The second line of the input contains n integers $h_{1,1}, h_{1,2}, \ldots, h_{1,n}$ ($1 \le h_{1,i} \le 10^9$), where $h_{1,i}$ is the height of the i-th student in the first row.

The third line of the input contains n integers $h_{2,1}, h_{2,2}, \ldots, h_{2,n}$ ($1 \le h_{2,i} \le 10^9$), where $h_{2,i}$ is the height of the i-th student in the second row.

Output

Print a single integer — the maximum possible total height of players in a team Demid can choose.

Examples

input

```
5
9 3 5 7 3
5 8 1 4 5
```

output

```
29
```

input

```
3
1 2 9
10 1 1
```

output

```
19
```

input

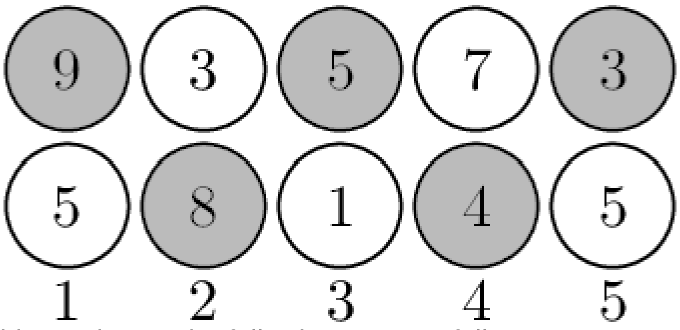
```
1
7
4
```

output

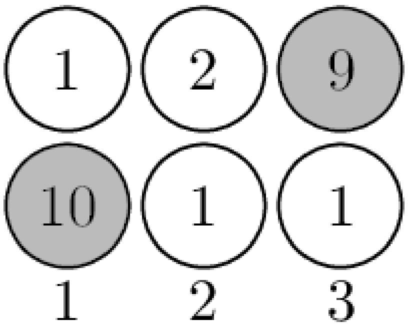
```
7
```

Note

In the first example Demid can choose the following team as follows:



In the second example Demid can choose the following team as follows:



G. Game 23

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp plays "Game 23". Initially he has a number n and his goal is to transform it to m. In one move, he can multiply n by 2 or multiply n by 3. He can perform any number of moves.

Print the number of moves needed to transform n to m. Print `-1` if it is impossible to do so.

It is easy to prove that any way to transform n to m contains the same number of moves (i.e. number of moves doesn't depend on the way of transformation).

Input

The only line of the input contains two integers n and m ($1 \le n \le m \le 5 \cdot 10^8$).

Output

Print the number of moves to transform n to m, or `-1` if there is no solution.

Examples

input

```
120 51840
```

output

```
7
```

input

```
42 42
```

output

```
0
```

input

```
48 72
```

output

```
-1
```

Note

In the first example, the possible sequence of moves is: $120 \rightarrow 240 \rightarrow 720 \rightarrow 1440 \rightarrow 4320 \rightarrow 12960 \rightarrow 25920 \rightarrow 51840.$ The are 7 steps in total.

In the second example, no moves are needed. Thus, the answer is 0.

In the third example, it is impossible to transform 48 to 72.

H. Chat room

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya has recently learned to type and log on to the Internet. He immediately entered a chat room and decided to say hello to everybody. Vasya typed the word s. It is considered that Vasya managed to say hello if several letters can be deleted from the typed word so that it resulted in the word "`hello`". For example, if Vasya types the word "`ahhellllloou`", it will be considered that he said hello, and if he types "`hlelo`", it will be considered that Vasya got misunderstood and he didn't manage to say hello. Determine whether Vasya managed to say hello by the given word s.

Input

The first and only line contains the word s, which Vasya typed. This word consisits of small Latin letters, its length is no less that 1 and no more than 100 letters.

Output

If Vasya managed to say hello, print "YES", otherwise print "NO".

Examples

input

```
ahhellllloou
```

output

```
YES
```

input

```
hlelo
```

output

```
NO
```

I. Lucky Pascal Triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Do you know what a Pascal Triangle is? No? Need a refresher? Then here are the key points:

- Pascal Triangle is made of multiple rows, starting from 0.
- The i-th row has $i + 1$ elements.
- Each row starts and ends with 1.
- We can consider the Pascal Triangle to be a 2D array, where `pascal[i][j]` gives us the value of the element in i-th row and j-th column of the Pascal Triangle.
- Except for the first and last column of each row, all other columns can be calculated as `pascal[i][j]` = `pascal[i-1][j]` + `pascal[i-1][j-1]`.

For example, here are the first 10 rows of Pascal Triangle.

```
                              1
                          1       1
                      1       2       1
                  1       3       3       1
              1       4       6       4       1
          1       5      10      10       5       1
      1       6      15      20      15       6       1
   1      7      21      35      35      21       7      1
 1     8     28      56      70      56      28     8     1
1    9    36     84     126     126     84     36     9    1
```

Ok. Now that we have revised what a Pascal Triangle is, let us talk about a new kind Pascal Triangle, called "Lucky Pascal Triangle". Legend says that a Lucky Pascal Triangle is a Pascal Triangle containing only elements that are divisible by the ultimate lucky number 7. All other elements that are not divisible by 7 are converted to 0. So, the first 10 rows of a Lucky Pascal Triangle will be:

```
                              0
                          0       0
                      0       0       0
                  0       0       0       0
              0       0       0       0       0
          0       0       0       0       0       0
      0       0       0       0       0       0       0
   0      7      21      35      35      21       7      0
 0     0     28      56      70      56      28     0     0
0    0    0     84     126     126     84     0     0    0
```

Fascinating right? It seems like there is some kind of pattern in the Lucky Pascal Triangle. We need to investigate more.

As the first step of the investigation, we need to know how many non-zero elements there are in a Lucky Pascal Triangle with N rows. Can you please help us?

Since the answer can be huge, please output your result modulo $10^9 + 7$.

Input

The first line of the input contains a single integer T ($1 \le T \le 10^5$) denoting the number of test cases. Next T lines follow with a single non-negative integer N ($N \le 10^{18}$).

Output

For each value of N, print the test case number and output the number of non-zero elements in the Lucky Pascal Triangle with N rows, modulo $10^9 + 7$. See sample input/output for more details

Example

standard input	standard output

`5`
`1`
`5`
`10`
`15`
`100000`
	`Case 1: 0`
	`Case 2: 0`
	`Case 3: 18`
	`Case 4: 43`
	`Case 5: 653881477`

J. String Problem

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

Boy Valera likes strings. And even more he likes them, when they are identical. That's why in his spare time Valera plays the following game. He takes any two strings, consisting of lower case Latin letters, and tries to make them identical. According to the game rules, with each move Valera can change **one** arbitrary character A_i in one of the strings into arbitrary character B_i, but he has to pay for every move a particular sum of money, equal to W_i. He is allowed to make as many moves as he needs. Since Valera is a very economical boy and never wastes his money, he asked you, an experienced programmer, to help him answer the question: what minimum amount of money should Valera have to get identical strings.

Input

The first input line contains two initial non-empty strings s and t, consisting of lower case Latin letters. The length of each string doesn't exceed 10^5. The following line contains integer n ($0 \le n \le 500$) — amount of possible changings. Then follow n lines, each containing characters A_i and B_i (lower case Latin letters) and integer W_i ($0 \le W_i \le 100$), saying that it's allowed to change character A_i into character B_i in any of the strings and spend sum of money W_i.

Output

If the answer exists, output the answer to the problem, and the resulting string. Otherwise output `-1` in the only line. If the answer is not unique, output any.

Examples

input

```
uayd
uxxd
3
a x 8
x y 13
d c 3
```

output

```
21
uxyd
```

input

```
a
b
3
a b 2
a b 3
b a 5
```

output

```
2
b
```

input

```
abc
ab
6
a b 4
```

```
a b 7
b a 8
c b 11
c a 3
a c 0
```

output

```
-1
```

K. Lucky Array

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

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.

Petya has an array consisting of n numbers. He wants to perform m operations of two types:

- add l r d — add an integer d to all elements whose indexes belong to the interval from l to r, inclusive $(1 \le l \le r \le n, 1 \le d \le 10^4)$;
- count l r — find and print on the screen how many lucky numbers there are among elements with indexes that belong to the interval from l to r inclusive $(1 \le l \le r \le n)$. Each lucky number should be counted as many times as it appears in the interval.

Petya has a list of all operations. The operations are such that after all additions the array won't have numbers that would exceed 10^4. Help Petya write a program that would perform these operations.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ — the number of numbers in the array and the number of operations correspondingly. The second line contains n positive integers, none of which exceeds 10^4 — those are the array numbers. Next m lines contain operations, one per line. They correspond to the description given in the statement.

It is guaranteed that after all operations are fulfilled each number in the array will not exceed 10^4.

Output

For each operation of the second type print the single number on the single line — the number of lucky numbers in the corresponding interval.

Examples

input

```
3 6
2 3 4
count 1 3
count 1 2
add 1 3 2
count 1 3
add 2 3 3
count 1 3
```

output

```
1
0
1
1
```

input

```
4 5
4 4 4 4
count 1 4
add 1 4 3
```

```
count 1 4
add 2 3 40
count 1 4
```

output

```
4
4
4
```

Note

In the first sample after the first addition the array will look in the following manner:

```
4 5 6
```

After the second addition:

```
4 8 9
```

The second sample after the first addition:

```
7 7 7 7
```

After the second addition:

```
7 47 47 7
```

L. Mysterious Present

time limit per test: 1 second
memory limit per test: 64 megabytes
input: standard input
output: standard output

Peter decided to wish happy birthday to his friend from Australia and send him a card. To make his present more mysterious, he decided to make a *chain*. Chain here is such a sequence of envelopes $A = \{a_1, a_2, ..., a_n\}$, where the width and the height of the i-th envelope is strictly higher than the width and the height of the $(i - 1)$-th envelope respectively. Chain size is the number of envelopes in the chain.

Peter wants to make the chain of the maximum size from the envelopes he has, the chain should be such, that he'll be able to put a card into it. The card fits into the chain if its width and height is lower than the width and the height of the smallest envelope in the chain respectively. It's forbidden to turn the card and the envelopes.

Peter has very many envelopes and very little time, this hard task is entrusted to you.

Input

The first line contains integers n, w, h $(1 \le n \le 5000, 1 \le w, h \le 10^6)$ — amount of envelopes Peter has, the card width and height respectively. Then there follow n lines, each of them contains two integer numbers w_i and h_i — width and height of the i-th envelope $(1 \le w_i, h_i \le 10^6)$.

Output

In the first line print the maximum chain size. In the second line print the numbers of the envelopes (separated by space), forming the required chain, starting with the number of the smallest envelope. Remember, please, that the card should fit into the smallest envelope. If the chain of maximum size is not unique, print any of the answers.

If the card does not fit into any of the envelopes, print number 0 in the single line.

Examples

input

```
2 1 1
2 2
2 2
```

output

```
1
1
```

input

```
3 3 3
5 4
12 11
9 8
```

output

```
3
1 3 2
```