Codeforces Round #653 (Div. 3)

A. Required Remainder

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

You are given three integers and . Your task is to find the **maximum** integer such that that , where is modulo operation. Many programming languages use percent operator % to implement it.

In other words, with given and you need to find the maximum possible integer from to that has the remainder modulo .

You have to answer independent test cases. It is guaranteed that such exists for each test case.

Input

The first line of the input contains one integer ( ) — the number of test cases. The next lines contain test cases.

The only line of the test case contains three integers and ( ).

It can be shown that such always exists under the given constraints.

Output

For each test case, print the answer — **maximum non-negative** integer such that and . It is guaranteed that the answer always exists.

Example

input
7 7 5 12345 5 0 4 10 5 15 17 8 54321 499999993 9 1000000000 10 5 187 2 0 999999999
output
12339 0 15 54306 999999995 185 999999998

Note

In the first test case of the example, the answer is (thus,). It is obvious that there is no greater integer not exceeding which has the remainder modulo .

B. Multiply by 2, divide by 6

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

You are given an integer . In one move, you can either multiply by two or divide by (if it is divisible by without the remainder).

Your task is to find the minimum number of moves needed to obtain from or determine if it's impossible to do that.

You have to answer independent test cases.

Input

The first line of the input contains one integer ( ) — the number of test cases. Then test cases follow.

The only line of the test case contains one integer ( ).

Output

For each test case, print the answer — the minimum number of moves needed to obtain from if it's possible to do that or -1 if

it's impossible to obtain from .

Example

input
7 1 2 3 12 12345 15116544 387420489
output
0 -1 2 -1 -1 12 36

Note

Consider the sixth test case of the example. The answer can be obtained by the following sequence of moves from the given integer :

1. Divide by and get ;
2. divide by and get ;
3. divide by and get ;
4. divide by and get ;
5. multiply by and get ;
6. divide by and get ;
7. divide by and get ;
8. divide by and get ;
9. multiply by and get ;
10. divide by and get ;
11. divide by and get ;
12. divide by and get .

C. Move Brackets

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

You are given a bracket sequence of length , where is even (divisible by two). The string consists of — opening brackets '(' and — closing brackets ')'.

In one move, you can choose **exactly one bracket** and move it to the beginning of the string or to the end of the string (i.e. you choose some index , remove the -th character of and insert it before or after all remaining characters of).

Your task is to find the minimum number of moves required to obtain **regular bracket sequence** from . It can be proved that the answer always exists under the given constraints.

Recall what the regular bracket sequence is:

- "()" is regular bracket sequence;
- if is regular bracket sequence then "(" + + ")" is regular bracket sequence;
- if and are regular bracket sequences then + is regular bracket sequence.

For example, "()()", "(())()", "(())" and "()" are regular bracket sequences, but ")(", "()(" and ")))" are not.

You have to answer independent test cases.

Input

The first line of the input contains one integer ( ) — the number of test cases. Then test cases follow.

The first line of the test case contains one integer ( ) — the length of . It is guaranteed that is even. The second line of the test case containg the string consisting of — opening and — closing brackets.

Output

For each test case, print the answer — the minimum number of moves required to obtain **regular bracket sequence** from . It can be proved that the answer always exists under the given constraints.

Example

input

```
4
2
)(
4
()()
8
())()()(
10
)))((((())
```

output

```
1
0
1
3
```

Note

In the first test case of the example, it is sufficient to move the first bracket to the end of the string.

In the third test case of the example, it is sufficient to move the last bracket to the beginning of the string.

In the fourth test case of the example, we can choose last three openning brackets, move them to the beginning of the string and obtain "((( ))) (( ))".

D. Zero Remainder Array

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

You are given an array consisting of positive integers.

Initially, you have an integer . During one move, you can do one of the following two operations:

1. Choose **exactly one** from to and increase by ( ), then increase by ( ).
2. Just increase by ( ).

The first operation can be applied **no more than once** to each from to .

Your task is to find the minimum number of moves required to obtain such an array that each its element is **divisible by** (the value is given).

You have to answer independent test cases.

Input

The first line of the input contains one integer ( ) — the number of test cases. Then test cases follow.

The first line of the test case contains two integers and ( ) — the length of and the required divisior. The second line of the test case contains integers ( ), where is the -th element of .

It is guaranteed that the sum of does not exceed ( ).

Output

For each test case, print the answer — the minimum number of moves required to obtain such an array that each its element is **divisible by** .

Example

input

```
5
4 3
1 2 1 3
10 6
8 7 1 8 3 7 5 10 8 9
5 10
20 100 50 20 100500
10 25
24 24 24 24 24 24 24 24 24 24
8 8
1 2 3 4 5 6 7 8
```

output

```
6
18
0
227
8
```

Note

Consider the first test case of the example:

1. , . Just increase ;
2. , . Add to the second element and increase ;
3. , . Add to the third element and increase ;
4. , . Add to the fourth element and increase ;
5. , . Just increase ;
6. , . Add to the first element and increase ;
7. , . We obtained the required array.

Note that you can't add to the same element more than once.

E1. Reading Books (easy version)

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

Easy and hard versions are actually different problems, so read statements of both problems completely and carefully.

Summer vacation has started so Alice and Bob want to play and joy, but... Their mom doesn't think so. She says that they have to read some amount of books before all entertainments. Alice and Bob will read each book **together** to end this exercise faster.

There are books in the family library. The -th book is described by three integers: — the amount of time Alice and Bob need to spend to read it, (equals if Alice likes the -th book and if not), and (equals if Bob likes the -th book and if not).

So they need to choose some books from the given books in such a way that:

- Alice likes **at least** books from the chosen set and Bob likes **at least** books from the chosen set;
- the total reading time of these books is **minimized** (they are children and want to play and joy as soon a possible).

The set they choose is **the same** for both Alice an Bob (it's shared between them) and they read all books **together**, so the total reading time is the sum of over all books that are in the chosen set.

Your task is to help them and find any suitable set of books or determine that it is impossible to find such a set.

Input

The first line of the input contains two integers and ( ).

The next lines contain descriptions of books, one description per line: the -th line contains three integers , and (,), where:

- — the amount of time required for reading the -th book;
- equals if Alice likes the -th book and otherwise;
- equals if Bob likes the -th book and otherwise.

Output

If there is no solution, print only one integer -1. Otherwise print one integer — the minimum total reading time of the suitable set of books.

Examples

input
8 4 7 1 1 2 1 1 4 0 1 8 1 1 1 0 1 1 1 1 1 0 1 3 0 0
output
18

input
5 2 6 0 0 9 0 0 1 0 1 2 1 1 5 1 0
output
8

input

```
5 3
3 0 0
2 1 0
3 1 0
5 0 1
3 0 1
```

output

```
-1
```

E2. Reading Books (hard version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Easy and hard versions are actually different problems, so read statements of both problems completely and carefully.

Summer vacation has started so Alice and Bob want to play and joy, but... Their mom doesn't think so. She says that they have to read **exactly** books before all entertainments. Alice and Bob will read each book **together** to end this exercise faster.

There are books in the family library. The -th book is described by three integers: — the amount of time Alice and Bob need to spend to read it, (equals if Alice likes the -th book and if not), and (equals if Bob likes the -th book and if not).

So they need to choose **exactly** books from the given books in such a way that:

- Alice likes **at least** books from the chosen set and Bob likes **at least** books from the chosen set;
- the total reading time of these books is **minimized** (they are children and want to play and joy as soon a possible).

The set they choose is **the same** for both Alice an Bob (it's shared between them) and they read all books **together**, so the total reading time is the sum of over all books that are in the chosen set.

Your task is to help them and find any suitable set of books or determine that it is impossible to find such a set.

Input

The first line of the input contains three integers , and ( ).

The next lines contain descriptions of books, one description per line: the -th line contains three integers , and (,), where:

- — the amount of time required for reading the -th book;
- equals if Alice likes the -th book and otherwise;
- equals if Bob likes the -th book and otherwise.

Output

If there is no solution, print only one integer -1.

If the solution exists, print in the first line — the minimum total reading time of the suitable set of books. In the second line print distinct integers from to in any order — indices of books which are in the set you found.

If there are several answers, print any of them.

Examples

input

```
6 3 1
6 0 0
11 1 0
9 0 1
21 1 1
10 1 0
8 0 1
```

output

```
24
6 5 1
```

input

```
6 3 2
6 0 0
11 1 0
9 0 1
21 1 1
10 1 0
8 0 1
```

output

```
39
4 6 5
```

F. Cyclic Shifts Sorting

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

You are given an array consisting of integers.

In one move, you can choose some index ( ) and shift the segment cyclically to the right (i.e. replace the segment with).

Your task is to sort the initial array by **no more than such operations** or say that it is impossible to do that.

You have to answer independent test cases.

Input

The first line of the input contains one integer ( ) — the number of test cases. Then test cases follow.

The first line of the test case contains one integer ( ) — the length of . The second line of the test case contains integers ( ), where is the -th element .

It is guaranteed that the sum of does not exceed .

Output

For each test case, print the answer: -1 on the only line if it is impossible to sort the given array using operations described in the problem statement, or the number of operations on the first line and integers ( ), where is the index of left border of the segment for the -th operation. You should print indices **in order of performing operations**.

Example

input
5 5 1 2 3 4 5 5 5 4 3 2 1 8 8 4 5 2 3 6 7 3 7 5 2 1 6 4 7 3 6 1 2 3 3 6 4
output
0 6 3 1 3 2 2 3 13 2 1 1 6 4 2 4 3 3 4 4 6 6 -1 4 3 3 4 4

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform