Deltix Round, Summer 2021 (open for everyone, rated, Div. 1 + Div. 2)

A. A Variety of Operations

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

William has two numbers a and b initially both equal to **zero**. William mastered performing three different operations with them quickly. Before performing each operation some positive integer k is picked, which is then used to perform one of the following operations: (note, that for each operation you can choose a **new** positive integer k)

1. add number k to both a and b, or
2. add number k to a and subtract k from b, or
3. add number k to b and subtract k from a.

Note that after performing operations, numbers a and b may become negative as well.

William wants to find out the minimal number of operations he would have to perform to make a equal to his favorite number c and b equal to his second favorite number d.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The only line of each test case contains two integers c and d ($0 \le c, d \le 10^9$), which are William's favorite numbers and which he wants a and b to be transformed into.

Output

For each test case output a single number, which is the minimal number of operations which William would have to perform to make a equal to c and b equal to d, or -1 if it is impossible to achieve this using the described operations.

Example

input
6 1 2 3 5 5 3 6 6 8 0 0 0
output
-1 2 2 1 2 0

Note

Let us demonstrate one of the suboptimal ways of getting a pair $(3, 5)$:

- Using an operation of the first type with $k = 1$, the current pair would be equal to $(1, 1)$.
- Using an operation of the third type with $k = 8$, the current pair would be equal to $(-7, 9)$.

- Using an operation of the second type with $k = 7$, the current pair would be equal to $(0, 2)$.
- Using an operation of the first type with $k = 3$, the current pair would be equal to $(3, 5)$.

B. Take Your Places!

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

William has an array of n integers $a_1, a_2, \ldots, a_n$. In one move he can swap two neighboring items. Two items a_i and a_j are considered neighboring if the condition $|i - j| = 1$ is satisfied.

William wants you to calculate the minimal number of swaps he would need to perform to make it so that the array does not contain two neighboring items with the same parity.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 10^5$) which is the total number of items in William's array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) which are William's array.

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

Output

For each test case output the minimal number of operations needed or -1 if it is impossible to get the array to a state when no neighboring numbers have the same parity.

Example

input
5 3 6 6 1 1 9 6 1 1 1 2 2 2 2 8 6 6 6 2 3 4 5 1
output
1 0 3 -1 2

Note

In the first test case the following sequence of operations would satisfy the requirements:

1. swap(2, 3). Array after performing the operation: $[6, 1, 6]$

In the second test case the array initially does not contain two neighboring items of the same parity.

In the third test case the following sequence of operations would satisfy the requirements:

1. swap(3, 4). Array after performing the operation: $[1, 1, 2, 1, 2, 2]$
2. swap(2, 3). Array after performing the operation: $[1, 2, 1, 1, 2, 2]$

3. swap(4, 5). Array after performing the operation: $[1, 2, 1, 2, 1, 2]$

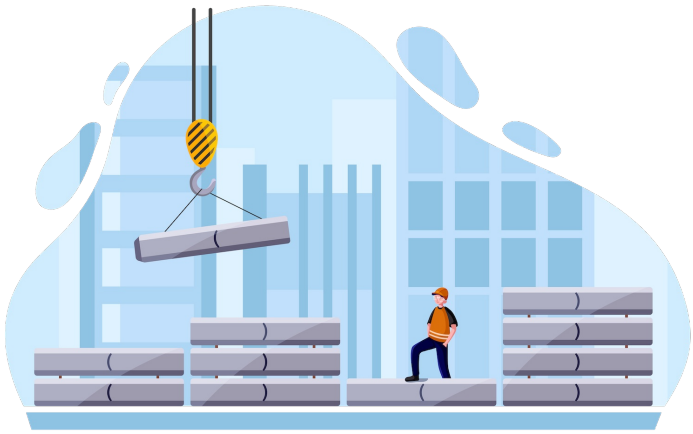
In the fourth test case it is impossible to satisfy the requirements.

In the fifth test case the following sequence of operations would satisfy the requirements:

1. swap(2, 3). Array after performing the operation: $[6, 3, 2, 4, 5, 1]$
2. swap(4, 5). Array after performing the operation: $[6, 3, 2, 5, 4, 1]$

C. Compressed Bracket Sequence

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



William has a favorite bracket sequence. Since his favorite sequence is quite big he provided it to you as a sequence of positive integers $c_1, c_2, \ldots, c_n$ where c_i is the number of consecutive brackets "(" if i is an odd number or the number of consecutive brackets ")" if i is an even number.

For example for a bracket sequence "((())()))" a corresponding sequence of numbers is $[3, 2, 1, 3]$.

You need to find the total number of continuous subsequences (subsegments) $[l, r]$ ($l \le r$) of the original bracket sequence, which are regular bracket sequences.

A bracket sequence is called regular if it is possible to obtain correct arithmetic expression by inserting characters "+" and "1" into this sequence. For example, sequences "((())())", "()" and "((()(())))" are regular, while ")(", "(()" and "((()))(" are not.

Input

The first line contains a single integer n $(1 \le n \le 1000)$, the size of the compressed sequence.

The second line contains a sequence of integers $c_1, c_2, \ldots, c_n$ $(1 \le c_i \le 10^9)$, the compressed sequence.

Output

Output a single integer — the total number of subsegments of the original bracket sequence, which are regular bracket sequences.

It can be proved that the answer fits in the signed 64-bit integer data type.

Examples

input
5 4 1 2 3 1
output
5

input
6 1 3 2 1 2 4
output
6

input
6 1 1 1 1 2 2
output
7

Note

In the first example a sequence (((()(()))(is described. This bracket sequence contains 5 subsegments which form regular bracket sequences:

1. Subsequence from the 3rd to 10th character: (()(()))
2. Subsequence from the 4th to 5th character: ()
3. Subsequence from the 4th to 9th character: ()(())
4. Subsequence from the 6th to 9th character: (())
5. Subsequence from the 7th to 8th character: ()

In the second example a sequence ()))(()((()))) is described.

In the third example a sequence ()()(()) is described.

D. Take a Guess

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

This is an interactive task

William has a certain sequence of integers $a_1, a_2, \ldots, a_n$ in his mind, but due to security concerns, he does not want to reveal it to you completely. William is ready to respond to no more than $2 \cdot n$ of the following questions:

- What is the result of a bitwise AND of two items with indices i and j ($i \neq j$)
- What is the result of a bitwise OR of two items with indices i and j ($i \neq j$)

You can ask William these questions and you need to find the k-th smallest number of the sequence.

Formally the k-th smallest number is equal to the number at the k-th place in a 1-indexed array sorted in non-decreasing order. For example in array $[5, 3, 3, 10, 1]$ 4th smallest number is equal to 5, and 2nd and 3rd are 3.

Input

It is guaranteed that for each element in a sequence the condition $0 \leq a_i \leq 10^9$ is satisfied.

Interaction

In the first line you will be given two integers n and k $(3 \leq n \leq 10^4, 1 \leq k \leq n)$, which are the number of items in the sequence a and the number k.

After that, you can ask no more than $2 \cdot n$ questions (not including the "`finish`" operation).

Each line of your output may be of one of the following types:

- "`or i j`" $(1 \leq i, j \leq n, i \neq j)$, where i and j are indices of items for which you want to calculate the bitwise OR.
- "`and i j`" $(1 \leq i, j \leq n, i \neq j)$, where i and j are indices of items for which you want to calculate the bitwise AND.
- "`finish res`", where res is the kth smallest number in the sequence. After outputting this line the program execution must conclude.

In response to the first two types of queries, you will get an integer x, the result of the operation for the numbers you have selected.

After outputting a line do not forget to output a new line character and flush the output buffer. Otherwise you will get the "`Idleness limit exceeded`". To flush the buffer use:

- `fflush(stdout)` in C++
- `System.out.flush()` in Java
- `stdout.flush()` in Python
- `flush(output)` in Pascal
- for other languages refer to documentation

If you perform an incorrect query the response will be -1. After receiving response -1 you must immediately halt your program in order to receive an "`Incorrect answer`" verdict.

Hacking

To perform a hack you will need to use the following format:

The first line must contain two integers n and k $(3 \le n \le 10^4, 1 \le k \le n)$, which are the number of items in the sequence a and the number k.

The second line must contain n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$, the sequence a.

Example

input
7 6 2 7
output
and 2 5 or 5 6 finish 5

Note

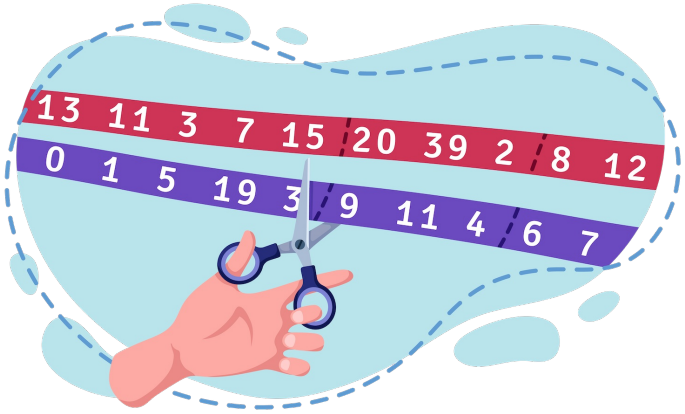
In the example, the hidden sequence is $[1, 6, 4, 2, 3, 5, 4]$.

Below is the interaction in the example.

Query (contestant's program)	Response (interactor)	Notes
and 2 5	2	$a_2 = 6$, $a_5 = 3$. Interactor returns bitwise AND of the given numbers.
or 5 6	7	$a_5 = 3$, $a_6 = 5$. Interactor returns bitwise OR of the given numbers.
finish 5		5 is the correct answer. Note that you must find the value and not the index of the kth smallest number.

E. Equilibrium

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



William has two arrays a and b, each consisting of n items.

For some segments $l..r$ of these arrays William wants to know if it is possible to equalize the values of items in these segments using a balancing operation. Formally, the values are equalized if for each i from l to r holds $a_i = b_i$.

To perform a balancing operation an even number of indices must be selected, such that $l \le pos_1 < pos_2 < \cdots < pos_k \le r$. Next the items of **array a** at positions $pos_1, pos_3, pos_5, \ldots$ get incremented by one and the items of **array b** at positions $pos_2, pos_4, pos_6, \ldots$ get incremented by one.

William wants to find out if it is possible to equalize the values of elements in two arrays for each segment using some number of

balancing operations, and what is the minimal number of operations required for that. Note that for each segment the operations are performed independently.

Input

The first line contains a two integers n and q ($2 \le n \le 10^5$, $1 \le q \le 10^5$), the size of arrays a and b and the number of segments.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$.

The third line contains n integers $b_1, b_2, \ldots, b_n$ $(0 \le b_i \le 10^9)$.

Each of the next q lines contains two integers l_i and r_i $(1 \le l_i < r_i \le n)$, the edges of segments.

Output

For each segment output a single number — the minimal number of balancing operations needed or "-1" if it is impossible to equalize segments of arrays.

Example

input
8 5 0 1 2 9 3 2 7 5 2 2 1 9 4 1 5 8 2 6 1 7 2 4 7 8 5 8
output
1 3 1 -1 -1

Note

For the first segment from 2 to 6 you can do one operation with $pos = [2, 3, 5, 6]$, after this operation the arrays will be: $a = [0, 2, 2, 9, 4, 2, 7, 5]$, $b = [2, 2, 2, 9, 4, 2, 5, 8]$. Arrays are equal on a segment from 2 to 6 after this operation.

For the second segment from 1 to 7 you can do three following operations:

1. $pos = [1, 3, 5, 6]$
2. $pos = [1, 7]$
3. $pos = [2, 7]$

After these operations, the arrays will be: $a = [2, 2, 2, 9, 4, 2, 7, 5]$, $b = [2, 2, 2, 9, 4, 2, 7, 8]$. Arrays are equal on a segment from 1 to 7 after these operations.

For the third segment from 2 to 4 you can do one operation with $pos = [2, 3]$, after the operation arrays will be: $a = [0, 2, 2, 9, 3, 2, 7, 5]$, $b = [2, 2, 2, 9, 4, 1, 5, 8]$. Arrays are equal on a segment from 2 to 4 after this operation.

It is impossible to equalize the fourth and the fifth segment.

F. Sports Betting

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

William is not only interested in trading but also in betting on sports matches. n teams participate in each match. Each team is characterized by strength a_i. Each two teams $i < j$ play with each other exactly once. Team i wins with probability $\frac{a_i}{a_i + a_j}$ and team

j wins with probability $\frac{a_j}{a_i+a_j}$.

The team is called a winner if it directly or indirectly defeated all other teams. Team a defeated (directly or indirectly) team b if there is a sequence of teams c_1, c_2, ... c_k such that $c_1 = a$, $c_k = b$ and team c_i defeated team c_{i+1} for all i from 1 to $k - 1$. Note that it is possible that team a defeated team b and in the same time team b defeated team a.

William wants you to find the expected value of the number of winners.

Input

The first line contains a single integer n ($1 \le n \le 14$), which is the total number of teams participating in a match.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the strengths of teams participating in a match.

Output

Output a single integer — the expected value of the number of winners of the tournament modulo $10^9 + 7$.

Formally, let $M = 10^9 + 7$. It can be demonstrated that the answer can be presented as a irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output a single integer equal to $p \cdot q^{-1} \bmod M$. In other words, output an integer x such that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Examples

input
2 1 2
output
1

input
5 1 5 2 11 14
output
642377629

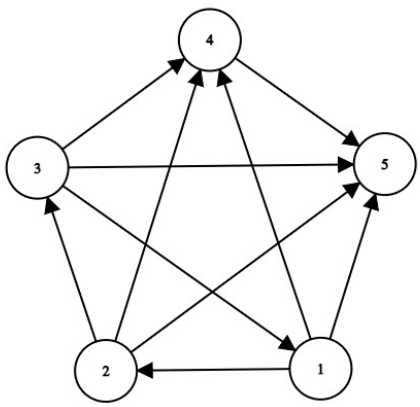
Note

To better understand in which situation several winners are possible let's examine the second test:

One possible result of the tournament is as follows ($a \to b$ means that a defeated b):

- $1 \to 2$
- $2 \to 3$
- $3 \to 1$
- $1 \to 4$
- $1 \to 5$
- $2 \to 4$
- $2 \to 5$
- $3 \to 4$
- $3 \to 5$
- $4 \to 5$

Or more clearly in the picture:



In this case every team from the set $\{1, 2, 3\}$ directly or indirectly defeated everyone. I.e.:

- 1st defeated everyone because they can get to everyone else in the following way $1 \to 2$, $1 \to 2 \to 3$, $1 \to 4$, $1 \to 5$.
- 2nd defeated everyone because they can get to everyone else in the following way $2 \to 3$, $2 \to 3 \to 1$, $2 \to 4$, $2 \to 5$.
- 3rd defeated everyone because they can get to everyone else in the following way $3 \to 1$, $3 \to 1 \to 2$, $3 \to 4$, $3 \to 5$.

Therefore the total number of winners is 3.

G. Gates to Another World

time limit per test: 4 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

As mentioned previously William really likes playing video games. In one of his favorite games, the player character is in a universe where every planet is designated by a binary number from 0 to $2^n - 1$. On each planet, there are gates that allow the player to move from planet i to planet j if the binary representations of i and j differ in exactly one bit.

William wants to test you and see how you can handle processing the following queries in this game universe:

- Destroy planets with numbers from l to r inclusively. These planets cannot be moved to anymore.
- Figure out if it is possible to reach planet b from planet a using some number of planetary gates. It is guaranteed that the planets a and b are not destroyed.

Input

The first line contains two integers n, m ($1 \leq n \leq 50$, $1 \leq m \leq 5 \cdot 10^4$), which are the number of bits in binary representation of each planets' designation and the number of queries, respectively.

Each of the next m lines contains a query of two types:

`block l r` — query for destruction of planets with numbers from l to r inclusively ($0 \leq l \leq r < 2^n$). It's guaranteed that no planet will be destroyed twice.

`ask a b` — query for reachability between planets a and b ($0 \leq a, b < 2^n$). It's guaranteed that planets a and b hasn't been destroyed yet.

Output

For each query of type `ask` you must output "`1`" in a new line, if it is possible to reach planet b from planet a and "`0`" otherwise (without quotation marks).

Examples

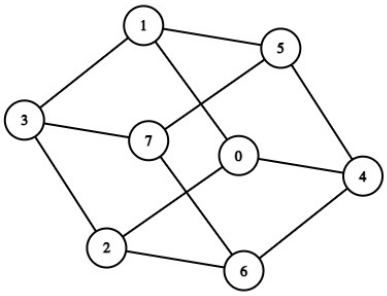
input
3 3 ask 0 7 block 3 6 ask 0 7
output
1 0

input
6 10 block 12 26 ask 44 63 block 32 46 ask 1 54 block 27 30 ask 10 31 ask 11 31 ask 49 31 block 31 31 ask 2 51
output
1 1

```
0
0
1
0
```

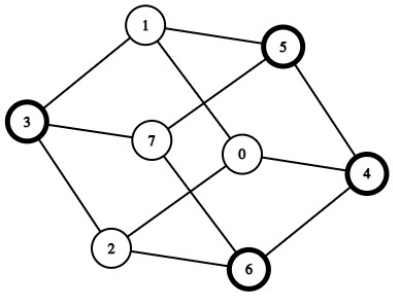
Note

The first example test can be visualized in the following way:



Response to a query `ask 0 7` is positive.

Next after query `block 3 6` the graph will look the following way (destroyed vertices are highlighted):



Response to a query `ask 0 7` is negative, since any path from vertex 0 to vertex 7 must go through one of the destroyed vertices.

H. DIY Tree

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

William really likes puzzle kits. For one of his birthdays, his friends gifted him a complete undirected edge-weighted graph consisting of n vertices.

He wants to build a spanning tree of this graph, such that for the first k vertices the following condition is satisfied: the degree of a vertex with index i does not exceed d_i. Vertices from $k+1$ to n may have any degree.

William wants you to find the minimum weight of a spanning tree that satisfies all the conditions.

A spanning tree is a subset of edges of a graph that forms a tree on all n vertices of the graph. The weight of a spanning tree is defined as the sum of weights of all the edges included in a spanning tree.

Input

The first line of input contains two integers n, k ($2 \le n \le 50$, $1 \le k \le min(n-1, 5)$).

The second line contains k integers $d_1, d_2, \ldots, d_k$ ($1 \le d_i \le n$).

The i-th of the next $n-1$ lines contains $n-i$ integers $w_{i,i+1}, w_{i,i+2}, \ldots, w_{i,n}$ ($1 \le w_{i,j} \le 100$): weights of edges $(i, i+1), (i, i+2), \ldots, (i, n)$.

Output

Print one integer: the minimum weight of a spanning tree under given degree constraints for the first k vertices.

Example

input
10 5 5 3 4 2 1 29 49 33 12 55 15 32 62 37 61 26 15 58 15 22 8 58 37 16 9 39 20 14 58 10 15 40 3 19 55 53 13 37 44 52 23 59 58 4 69 80 29 89 28 48
output
95