2019 LGE Code Jam: Online Round 2

Problem A. Array Rotation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 sec
Memory limit:	512 MB

You are given a two-dimensional, integer array X of size n by n (where n is odd).

You want to rotate some elements in X by a multiple of 45 degrees (either clockwise or counterclockwise). When you rotate X clockwise by 45 degrees, the following operations happen simultaneously:

- The n elements on X's main diagonal (i.e., at (1,1), (2,2), ..., (n,n)) are moved to the middle column (i.e., the (n+1)/2-th column) while preserving the order of elements.
- The n elements on X's middle column are moved to X's antidiagonal (i.e., at (n,1), (n-1,2), ..., (1,n)) while preserving the order of elements.
- The n elements on X's antidiagonal are moved to the middle row (i.e., the (n+1)/2-th row) while preserving the order of elements.
- The n elements on X's middle row are moved to X's main diagonal while preserving the order of elements.
- No other elements are moved during this operation.

Counterclockwise rotation by 45-degrees is defined analogously.

For instance, the following image shows an original array X (in the middle), and how it looks like after a rotation by 45 degrees in either direction – clockwise rotation on the right and counterclockwise rotation on the left.

1	2	**3**	4	**5**
6	**7**	**8**	**9**	10
11	**12**	**13**	**14**	**15**
16	**17**	**18**	**19**	20
21	22	**23**	24	**25**

Table 1: Array X (5x5)

3	2	**5**	4	**15**
6	**8**	**9**	**14**	10
1	**7**	**13**	**19**	**25**
16	**12**	**17**	**18**	20
11	22	**21**	24	**23**

Table 2: X rotated by 45 degrees counterclockwise.

11	2	**1**	4	**3**
6	**12**	**7**	**8**	10
21	**17**	**13**	**9**	**5**
16	**18**	**19**	**14**	20
23	22	**25**	24	**15**

Table 3: X rotated by 45 degrees clockwise.

Given an array X and a rotation parameter (which specifies by how much you wish to rotate X in which direction), output the values of X after this rotation happens.

Input

The first line will contain an integer T, the number of test cases ($1 \leq T \leq 10$).

For each test case: the first line will contain two integers n and d where $1 \leq n < 500$ (n is always odd) and $|d|$ is a multiple of 45 and $0 \leq |d| \leq 360$. When d is positive, it means X should be rotated clockwise by d degrees; otherwise, X should be rotated counterclockwise by $|d|$ degrees.

The next n lines will contain n integers each, specifying an element of X. Each element of X will be between 1 and 1,000,000, inclusive.

Output

For each test case: your output should describe X after you rotate X by d degrees. It should contain n lines where each line contains n integers separated by a whitespace (each line describes a row of X).

Examples

standard input	standard output
4 5 45 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 5 -45 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 5 135 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 5 360 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	11 2 1 4 3 6 12 7 8 10 21 17 13 9 5 16 18 19 14 20 23 22 25 24 15 3 2 5 4 15 6 8 9 14 10 1 7 13 19 25 16 12 17 18 20 11 22 21 24 23 23 2 21 4 11 6 18 17 12 10 25 19 13 7 1 16 14 9 8 20 15 22 5 24 3 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Problem B. Odd Subsequences

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 sec
Memory limit:	512 MB

A subsequence is a sequence that can be obtained from an array by deleting 0 or more elements. For a subsequence S of an array A with length n that consists of nonnegative integers, we define S as an "odd-subsequence" if it satisfies the following condition:

- Sum of elements of S has an odd number of its digits. The number of odd number(s) should be odd.

Given a nonnegative integer array A, output the number of distinct odd subsequences of A. To be precise, we treat two subsequences as equivalent if their sorted results are equal.

For instance, suppose $A =$ [3, 3, 6, 8, 6].

- Odd subsequences of A with length 0 : -
- Odd subsequences of A with length 1 : [3]
  - The sum of elements is 3, which has a single odd digit 3
- Odd subsequences of A with length 2 : [3, 6], [6, 6], [6, 8]
  - The sums of elements are 9, 12, 14 respectively, with single odd digit 9, 1, 1
- Odd subsequences of A with length 3 : [3, 3, 6], [3, 3, 8]
- Odd subsequences of A with length 4 : [3, 3, 6, 6], [3, 6, 6, 8]
- Odd subsequences of A with length 5 : -

Input

The first line will contain an integer T ($1 \leq T \leq 10$), the number of test cases.

For each test case: The first line will contain n, and the second line will contain n non-negative integers describing A. Each element of A will be between 0 and 2,000.

Output

For each test case, output in a slinge line the number of odd subsequences you can make out of A.

Subtask 1 (3 points)

- $1 \leq n \leq 10$

Subtask 2 (11 points)

- $1 \leq n \leq 60$

Examples

standard input	standard output
3 3 3 3 6 4 0 1 2 3 5 3 3 6 8 6	3 8 8

Notes

- Test case 1: 3, 3,6, and 3, 3, 6 are the only three odd subsequences of A.
- Test case 2: 1, 3, 0,1, 0,3, 1,2, 2,3, 0,1,2, and 0,2,3 are the odd subsequences of A.
- Test case 3: 8 odd s of A were described in the problem statement.

Problem C. Stock Market

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 sec
Memory limit:	512 MB

In this problem, your goal is to gain a profit from the stock market, with the following restrictions:

- The stock market opens for n days; from day 1 to day n.
- You can only buy stocks for consecutive k days within that period ($k \geq 1$). You must buy a stock everyday (for k consecutive days).
- The buying price for the stock is always fixed: It's y dollars per day.
- In contrast, the selling price for the stock depends on when you have bought it; It's $V[i]$ dollars if you have bought it on the day i.
- You start with X dollars as your initial budget.
- You cannot sell stocks before the market is completely closed (after day n).
- You must gain at least Z dollars of profit. ($Z \geq 0$)

Therefore, if you decide to buy stocks from the day i to the day j, with $1 \leq i \leq j \leq n$, then you must satisfy:

$(V[i] + V[i+1] + \cdots + V[j-1] + V[j]) - y \times (j - i + 1) \geq Z$ and $y \times (j - i + 1) \leq X$.

Your goal is to get the shortest period of which you are buying stocks meeting all requirements above;

In other words, maximizing the profit is not your interest,

instead, you are interested in minimizing the number of days of buying.

Return two integers i and j that define the stock buying period.

If there are multiple possible options, return the one that maximizes i.

(That is, return the one of which you enter the market as late as possible)

If there is no such possible option, return -1.

Input

The first line will contain the number of test cases T, ($1 \leq T|le10$).

For each test case, the first line will contain four integers, n, X, y, and Z. Assume $0 \leq X, y, Z \leq 10^9$.

The second line will contain n integers, representing $V[1], .., V[n]$, where $|V[i]|\, d10^6$.

Output

For each test, output a single line.

If no (i, j) can satisfy the said conditions, output -1.

Otherwise, among all possible (i, j)'s, find the one that minimizes $(j - i + 1)$ – if there are multiple such pairs, find the one that maximizes i.

Subtask 1 (5 points)

- $1 \leq n \leq 5,000$

Subtask 2 (11 points)

- $1 \leq n \leq 100,000$

Examples

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

Problem D. Tribal War

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 sec
Memory limit:	512 MB

There are N different tribes in a county, numbered from 1 to N.

- For any two different tribes, they are either in alliance or hostility. These relationships are symmetric.
- Note that even if tribe A and B are allies and B and C are allies, tribe A and C can be hostile to each other.
- For three different tribes, if all of them are in an alliance, we call them as in "3-way alliance"; Similarly, if all of them are in hostility, we call them as in "3-way hostility".

Given the relationships between N tribes, return the sum of all of the 3-way alliances and 3-way hostilities.

For instance, let $N = 5$ and (1, 2), (1, 3), and (4, 5) represent 3 alliance relationships (all other pairs are pairwise enemies). In this case (2, 3, 4) and (2, 3, 5) are "3-way hostility", and there is no "3-way alliance". Hence the answer is 2.

In another example, let $N = 5$ and (1, 2), (2, 3), and (1, 3) represent 3 alliance relationships (again, all others are pairwise enemies). In this case, (1, 2, 3) is a "3-way alliance" and (1, 4, 5), (2, 4, 5), and (3, 4, 5) are "3-way hostility". The answer is 4.

Input

The first line will contain the number of test cases, T ($1 \leq T \leq 10$).

For each test case, the first line will contain N and M separated by a whitespace.

For the next M lines, each line will contain two integers x and y such that tribes x and y are allies. You can assume that $1 \leq x < y \leq N$ for all M lines, and no duplicate pairs will be provided in the input. **Assume that all pairs of tribes that are not given in the input are pairwise enemies.**

Output

For each test case, output the 3-way AE Number in a single line.

Subtask 1 (4 points)

- $3 \leq N \leq 200$
- $1 \leq M \leq \min(1,000, N \times (N-1)/2)$

Subtask 2 (13 points)

- $3 \leq N \leq 5,00$
- $1 \leq M \leq \min(1,000,000, N \times (N-1)/2)$

Examples

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

Problem E. Carpool Matching

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 sec
Memory limit:	512 MB

You need to help your friend, Albert, who's developing a new carpool app. In particular, you need to design a matching algorithm for drivers and riders. For the initial version of the app, the service will be provided in selected regions only, which are located along a long horizontal road.

- Every driver and rider leave from "the city" that's located at the far left on the road. All other cities are located to the right of the city.
- There are N riders, and the i-th rider wants to get to a place that's x_i meters ($0 < x_i$) away from "the city" (that is, x_i meters to the right of "the city").
- There are M drivers, and the j-th driver is willing to give a ride to at most one rider whose destination is between y_j and z_j meters away from "the city", inclusive.

You need to meet these constraints while matching the most number of riders and drivers.

For instance, suppose that there are 3 riders and 3 drivers with x, y, z values as follows:

- $x = [10, 20, 30]$
- $y = [8, 2, 25]$
- $z = [8, 18, 35]$

In this case, the riders want to get to other cities that are 10m, 20m, and 30m away from "the city" while driver 1 is willing to give a ride to someone heading to a place exactly 8m away, driver 2 is willing to give a ride if someone's going to a place that is 2m or further away but no more than 18m away, and driver 3 with a 25m-35m range.

In this case, we can match (rider 1, driver2) and (rider 3, driver 3) – there is no way to match more than two pairs.

Given N, M, and x, y, z values, compute the maximum number of rider-driver pairs you can match while meeting the said constraints.

Input

The first line will contain the number of test cases, T ($1 \leq T \leq 10$).

For each test case, the first line will contain two integers N and M.

The next line will contain N integers separated by whitespace, representing x_i's ($1 \leq x_i \leq 1,000,000,000$).

The next M lines will contain two integers each, representing y_j and z_j ($1 \leq y_j \leq z_j \leq 1,000,000,000$).

Output

For each test case, output in a single line the maximum number of rider-driver pairs you can match.

Subtask 1 (7 points)

- $1 \leq N, M \leq 100$

Subtask 2 (13 points)

- $1 \leq N, M \leq 100,000$

Examples

standard input	standard output
3 3 3 10 20 30 8 8 2 18 25 35 4 4 2 3 4 5 1 4 2 4 3 5 3 4 3 3 1 2 3 10 20 30 40 50 60	2 4 0

Problem F. Binary string

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 sec
Memory limit:	512 MB

Given a binary string S consists of only 0 and 1, suppose the index of S is 1-based and denote $S[a : b]$ as a substring of S from indices a to b (inclusive).

We apply an operation to this string m times, which inserts a substring of the given string after applying bitwise inversion to the said substring.

The details are as follows:

- Each "insert" operation takes two integer parameters x and y (which can vary from operation to operation), which define the range.
- The 0-1 inversion of the substring $S[x : y]$ is to be inserted into S.
- The position to be inserted is right after $S[y]$. The length of S increases by $y - x + 1$ afterward.
- After each operation, S will be mutated; in other words, before the second operation, we start with the different string S that has been changed by the first operation.

For instance, suppose $S = 01010110$ and $n = 8, m = 2, k = 12$, and we apply two operations with $(x = 2, y = 4)$ and $(x = 3, y = 5)$.

- When you apply the first operation, you would first compute the inversion of $S[2 : 4] = 101$, and insert it into S to obtain $S = 01010100110$ (the bold text highlights the inserted string, after the inversion of 101).
- When you apply the second operation, you would first compute the inversion of $S[3 : 5] = 010$, and insert it into S to obtain $S = 01010101100110$ (the bold text highlights the inserted string, after the inversion of 010).
- Because $k = 12$, you conclude that the final answer is "010101011001".

Return the first k characters of the final output string obtained after m operations are applied on S.

Input

The first line will contain the number of test cases, T ($1 \leq T \leq 10$).

For each test case, the first line will contain three integers n, m, and k.

The second line will contain a 0-1 string of length n (each character is either 0 or 1).

The next m lines will contain two integers each, representing $x[i]$ and $y[i]$.

You may assume that $1 \leq x[i] \leq y[i] \leq$ length of S right before applying operation i.

Likewise, k will be no greater than the smaller of 10^6 and the final length of S (after m operations).

Output

For each test case, output a 0-1 string of length k, which is $S[1 : k]$ after you apply m operations on the initial input string S.

Subtask 1 (5 points)

- $1 \leq n \leq 10^3, 1 \leq m \leq 10^3, 1 \leq k \leq 10^5$

Subtask 2 (23 points)

- $1 \leq n \leq 10^6, 1 \leq m \leq 10^4, 1 \leq k \leq 10^6$

Examples

standard input	standard output
3 8 2 12 01010110 2 4 3 5 4 1 4 1010 1 1 8 2 16 10101100 1 2 2 8	010101011001 1001 1001101111001000