Codeforces Round 971 (Div. 4)

A. Minimize!

1 second, 256 megabytes

You are given two integers a and b ($a \leq b$). Over all possible integer values of c ($a \leq c \leq b$), find the minimum value of $(c - a) + (b - c)$.

Input

The first line contains t ($1 \leq t \leq 55$) — the number of test cases.

Each test case contains two integers a and b ($1 \leq a \leq b \leq 10$).

Output

For each test case, output the minimum possible value of $(c - a) + (b - c)$ on a new line.

input

```
3
1 2
3 10
5 5
```

output

```
1
7
0
```

In the first test case, you can choose $c = 1$ and obtain an answer of $(1 - 1) + (2 - 1) = 1$. It can be shown this is the minimum value possible.

In the second test case, you can choose $c = 6$ and obtain an answer of $(6 - 3) + (10 - 6) = 7$. It can be shown this is the minimum value possible.

B. osu!mania

1 second, 256 megabytes

You are playing your favorite rhythm game, osu!mania. The layout of your beatmap consists of n rows and 4 columns. Because notes at the bottom are closer, you will process the bottommost row first and the topmost row last. Each row will contain exactly one note, represented as a '#'.

For each note $1, 2, \ldots, n$, in the order of processing, output the column in which the note appears.

Input

The first line contains t ($1 \leq t \leq 100$) — the number of test cases.

For each test case, the first line contains n ($1 \leq n \leq 500$) — the number of rows of the beatmap.

The following n lines contain 4 characters. The i-th line represents the i-th row of the beatmap from the top. It is guaranteed that the characters are either '.' or '#', and exactly one of the characters is '#'.

It is guaranteed that the sum of n over all test cases does not exceed 500.

Output

For each test case, output n integers on a new line, the column that the i-th note appears in for all i from 1 to n.

input

```
3
4
#...
.#..
..#.
...#
2
.#..
.#..
1
...#
```

output

```
4 3 2 1
2 2
4
```

C. The Legend of Freya the Frog

2 seconds, 256 megabytes

Freya the Frog is traveling on the 2D coordinate plane. She is currently at point $(0, 0)$ and wants to go to point (x, y). In one move, she chooses an integer d such that $0 \leq d \leq k$ and jumps d spots forward in the direction she is facing.

Initially, she is facing the positive x direction. After every move, she will alternate between facing the positive x direction and the positive y direction (i.e., she will face the positive y direction on her second move, the positive x direction on her third move, and so on).

What is the minimum amount of moves she must perform to land on point (x, y)?

Input

The first line contains an integer t ($1 \leq t \leq 10^4$) — the number of test cases.

Each test case contains three integers x, y, and k ($0 \leq x, y \leq 10^9, 1 \leq k \leq 10^9$).

Output

For each test case, output the number of jumps Freya needs to make on a new line.

input

```
3
9 11 3
0 10 8
1000000 100000 10
```

output

```
8
4
199999
```

In the first sample, one optimal set of moves is if Freya jumps in the following way: $(0, 0) \to (2, 0) \to (2, 2) \to (3, 2) \to (3, 5) \to (6, 5) \to (6, 8) \to (9, 8) \to (9, 11)$. This takes 8 jumps.

D. Satyam and Counting

2 seconds, 256 megabytes

Satyam is given n distinct points on the 2D coordinate plane. **It is guaranteed that $0 \leq y_i \leq 1$ for all given points (x_i, y_i).** How many different nondegenerate right triangles* can be formed from choosing three different points as its vertices?

Two triangles a and b are different if there is a point v such that v is a vertex of a but not a vertex of b.

*A nondegenerate right triangle has positive area and an interior 90° angle.

Input

The first line contains an integer t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains an integer n ($3 \leq n \leq 2 \cdot 10^5$) — the number of points.

The following n lines contain two integers x_i and y_i ($0 \leq x_i \leq n$, $0 \leq y_i \leq 1$) — the i'th point that Satyam can choose from. It is guaranteed that all (x_i, y_i) are pairwise distinct.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

Output an integer for each test case, the number of distinct nondegenerate right triangles that can be formed from choosing three points.

input

```
3
5
1 0
1 1
3 0
5 0
2 1
3
0 0
1 0
3 0
9
1 0
2 0
3 0
4 0
5 0
2 1
7 1
8 1
9 1
```

output

```
4
0
8
```

The four triangles in question for the first test case:

E. Klee's SUPER DUPER LARGE Array!!!

2 seconds, 256 megabytes

Klee has an array a of length n containing integers $[k, k+1, \ldots, k+n-1]$ in that order. Klee wants to choose an index i ($1 \leq i \leq n$) such that $x = |a_1 + a_2 + \cdots + a_i - a_{i+1} - \cdots - a_n|$ is minimized. Note that for an arbitrary integer z, $|z|$ represents the absolute value of z.

Output the minimum possible value of x.

Input

The first line contains t ($1 \leq t \leq 10^4$) — the number of test cases.

Each test case contains two integers n and k ($2 \leq n, k \leq 10^9$) — the length of the array and the starting element of the array.

Output

For each test case, output the minimum value of x on a new line.

input

```
4
2 2
7 2
5 3
1000000000 1000000000
```

output

```
1
5
1
347369930
```

In the first sample, $a = [2, 3]$. When $i = 1$ is chosen, $x = |2 - 3| = 1$. It can be shown this is the minimum possible value of x.

In the third sample, $a = [3, 4, 5, 6, 7]$. When $i = 3$ is chosen, $x = |3 + 4 + 5 - 6 - 7| = 1$. It can be shown this is the minimum possible value of x.

F. Firefly's Queries

2 seconds, 256 megabytes

Firefly is given an array a of length n. Let c_i denote the i'th cyclic shift* of a. She creates a new array b such that $b = c_1 + c_2 + \cdots + c_n$ where $+$ represents concatenation†.

Then, she asks you q queries. For each query, output the sum of all elements in the subarray of b that starts from the l-th element and ends at the r-th element, inclusive of both ends.

*The x-th ($1 \leq x \leq n$) cyclic shift of the array a is $a_x, a_{x+1} \ldots a_n, a_1, a_2 \ldots a_{x-1}$. Note that the 1-st shift is the initial a.

†The concatenation of two arrays p and q of length n (in other words, $p + q$) is $p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_n$.

Input

The first line contains t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains two integers n and q ($1 \leq n, q \leq 2 \cdot 10^5$) — the length of the array and the number of queries.

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^6$).

The following q lines contain two integers l and r ($1 \leq l \leq r \leq n^2$) — the left and right bounds of the query.

Note that the large inputs may require the use of 64-bit integers.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ and the sum of q does not exceed $2 \cdot 10^5$.

Output

For each query, output the answer on a new line.

input

```
5
3 3
1 2 3
1 9
3 5
8 8
5 5
4 8 3 2 4
1 14
3 7
7 10
2 11
1 25
1 1
6
1 1
5 7
3 1 6 10 4
3 21
6 17
2 2
1 5
1 14
9 15
12 13
5 3
4 9 10 10 1
20 25
3 11
20 22
```

output

```
18
8
1
55
20
13
41
105
6
96
62
1
24
71
31
14
44
65
15
```

For the first test case, $b = [1, 2, 3, 2, 3, 1, 3, 1, 2]$.

G1. Yunli's Subarray Queries (easy version)

3 seconds, 512 megabytes

This is the easy version of the problem. In this version, it is guaranteed that $r = l + k - 1$ for all queries.

For an arbitrary array b, Yunli can perform the following operation any number of times:

- Select an index i. Set $b_i = x$ where x is any integer she desires (x is not limited to the interval $[1, n]$).

Denote $f(b)$ as the minimum number of operations she needs to perform until there exists a consecutive subarray* of length at least k in b.

Yunli is given an array a of size n and asks you q queries. In each query, you must output $\sum_{j=l+k-1}^{r} f([a_l, a_{l+1}, \ldots, a_j])$. Note that in this version, you are only required to output $f([a_l, a_{l+1}, \ldots, a_{l+k-1}])$.

* If there exists a consecutive subarray of length k that starts at index i ($1 \le i \le |b| - k + 1$), then $b_j = b_{j-1} + 1$ for all $i < j \le i + k - 1$.

Input

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

The first line of each test case contains three integers n, k, and q ($1 \le k \le n \le 2 \cdot 10^5$, $1 \le q \le 2 \cdot 10^5$) — the length of the array, the length of the consecutive subarray, and the number of queries.

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The following q lines contain two integers l and r ($1 \le l \le r \le n$, $r = l + k - 1$) — the bounds of the query.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$ and the sum of q over all test cases does not exceed $2 \cdot 10^5$.

Output

Output $\sum_{j=l+k-1}^{r} f([a_l, a_{l+1}, \ldots, a_j])$ for each query on a new line.

input

```
3
7 5 3
1 2 3 2 1 2 3
1 5
2 6
3 7
8 4 2
4 3 1 1 2 4 3 2
3 6
2 5
5 4 2
4 5 1 2 3
1 4
2 5
```

output

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

In the first query of the first testcase, $b = [1, 2, 3, 2, 1]$. Yunli can make a consecutive subarray of length 5 in 2 moves:

- Set $b_4 = 4$
- Set $b_5 = 5$

After operations $b = [1, 2, 3, 4, 5]$.
In the second query of the first testcase, $b = [2, 3, 2, 1, 2]$. Yunli can make a consecutive subarray of length 5 in 3 moves:

- Set $b_3 = 0$
- Set $b_2 = -1$
- Set $b_1 = -2$

After operations $b = [-2, -1, 0, 1, 2]$.

G2. Yunli's Subarray Queries (hard version)

3 seconds, 512 megabytes

This is the hard version of the problem. In this version, it is guaranteed that $r \ge l + k - 1$ for all queries.

For an arbitrary array b, Yunli can perform the following operation any number of times:

- Select an index i. Set $b_i = x$ where x is any integer she desires (x is not limited to the interval $[1, n]$).

Denote $f(b)$ as the minimum number of operations she needs to perform until there exists a consecutive subarray* of length at least k in b.

Yunli is given an array a of size n and asks you q queries. In each query, you must output $\sum_{j=l+k-1}^{r} f([a_l, a_{l+1}, \ldots, a_j])$.

* If there exists a consecutive subarray of length k that starts at index i ($1 \le i \le |b| - k + 1$), then $b_j = b_{j-1} + 1$ for all $i < j \le i + k - 1$.

Input

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

The first line of each test case contains three integers n, k, and q ($1 \le k \le n \le 2 \cdot 10^5$, $1 \le q \le 2 \cdot 10^5$) — the length of the array, the length of the consecutive subarray, and the number of queries.

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The following q lines contain two integers l and r ($1 \le l \le r \le n$, $r \ge l + k - 1$) — the bounds of the query.

It is guaranteed the sum of n over all test cases does not exceed $2 \cdot 10^5$ and the sum of q over all test cases does not exceed $2 \cdot 10^5$.

Output

Output $\sum_{j=l+k-1}^{r} f([a_l, a_{l+1}, \ldots, a_j])$ for each query on a new line.

input

```
3
7 5 3
1 2 3 2 1 2 3
1 7
2 7
3 7
8 4 2
4 3 1 1 2 4 3 2
3 6
1 5
5 4 2
4 5 1 2 3
1 4
1 5
```

output

```
6
5
2
5
2
2
5
2
3
```

In the second query of the first testcase, we calculate the following function values:

- $f([2, 3, 2, 1, 2]) = 3$ because Yunli can set $b_3 = 4$, $b_4 = 5$, and $b_5 = 6$, making a consecutive subarray of size 5 in 3 moves.
- $f([2, 3, 2, 1, 2, 3] = 2$ because we can set $b_3 = 0$ and $b_2 = -1$, making a consecutive subarray of size 5 in 2 moves (starting at position 2)

The answer to this query is $3 + 2 = 5$.

G3. Yunli's Subarray Queries (extreme version)

3 seconds, 512 megabytes

This is the extreme version of the problem. In this version, the output of each query is different from the easy and hard versions. It is also guaranteed that $r \ge l + k - 1$ for all queries.

For an arbitrary array b, Yunli can perform the following operation any number of times:

- Select an index i. Set $b_i = x$ where x is any integer she desires (x is not limited to the interval $[1, n]$).

Denote $f(b)$ as the minimum number of operations she needs to perform until there exists a consecutive subarray* of length at least k in b.

Yunli is given an array a of size n and asks you q queries. In each query, you must output $\sum_{i=l}^{r-k+1} \sum_{j=i+k-1}^{r} f([a_i, a_{i+1}, \ldots, a_j])$.

* If there exists a consecutive subarray of length k that starts at index i ($1 \le i \le |b| - k + 1$), then $b_j = b_{j-1} + 1$ for all $i < j \le i + k - 1$.

Input

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

The first line of each test case contains three integers n, k, and q ($1 \le k \le n \le 2 \cdot 10^5$, $1 \le q \le 2 \cdot 10^5$) — the length of the array, the length of the consecutive subarray, and the number of queries.

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The following q lines contain two integers l and r ($1 \le l \le r \le n$, $r \ge l + k - 1$) — the bounds of the query.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$ and the sum of q over all test cases does not exceed $2 \cdot 10^5$.

Output

Output $\sum_{i=l}^{r-k+1} \sum_{j=i+k-1}^{r} f([a_i, a_{i+1}, \ldots, a_j])$ for each query on a new line.

input

```
5
7 2 4
1 2 3 2 1 2 3
4 6
1 7
2 7
3 7
8 4 2
4 3 1 1 2 4 3 2
3 6
1 5
5 4 2
4 5 1 2 3
1 4
1 5
10 4 8
2 3 6 5 8 9 8 10 10 1
2 7
6 10
1 9
1 6
3 9
4 10
2 10
1 8
10 7 4
3 4 5 3 4 5 9 10 8 9
1 9
2 10
1 10
2 9
```

output

```
1
3
3
3
2
7
2
4
8
6
28
7
16
20
32
19
18
15
26
9
```

In the first query of the first testcase, we can calculate the answer for the query through the following:

- $i = 4$ and $j = 5$: $f([2, 1]) = 1$ because Yunli can set $b_2 = 3$, making a consecutive subarray of size 2 in 1 move.
- $i = 4$ and $j = 6$: $f([2, 1, 2]) = 0$ because there is already a consecutive array of size 2.
- $i = 5$ and $j = 6$: $f([1, 2]) = 0$ because there is already a consecutive array of size 2.

The answer to this query is $1 + 0 + 0 = 1$.