A. Find K Distinct Points with Fixed Center

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

I couldn't think of a good title for this problem, so I decided to learn from LeetCode.

— Sun Tzu, *The Art of War*

You are given three integers x_c, y_c, and k ($-100 \leq x_c, y_c \leq 100$, $1 \leq k \leq 1000$).

You need to find k **distinct** points (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k), having integer coordinates, on the 2D coordinate plane such that:

- their center* is (x_c, y_c)
- $-10^9 \leq x_i, y_i \leq 10^9$ for all i from 1 to k

It can be proven that at least one set of k distinct points always exists that satisfies these conditions.

*The center of k points (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k) is $\left(\frac{x_1+x_2+\ldots+x_k}{k}, \frac{y_1+y_2+\ldots+y_k}{k}\right)$.

Input

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

Each test case contains three integers x_c, y_c, and k ($-100 \leq x_c, y_c \leq 100$, $1 \leq k \leq 1000$) — the coordinates of the center and the number of distinct points you must output.

It is guaranteed that the sum of k over all test cases does not exceed 1000.

Output

For each test case, output k lines, the i-th line containing two space separated integers, x_i and y_i, ($-10^9 \leq x_i, y_i \leq 10^9$) — denoting the position of the i-th point.

If there are multiple answers, print any of them. It can be shown that a solution always exists under the given constraints.

Standard Input	Standard Output

4
10 10 1
0 0 3
-5 -8 8
4 -5 3
	10 10
	-1 -1
	5 -1
	-4 2
	-6 -7
	-5 -7
	-4 -7
	-4 -8
	-4 -9
	-5 -9
	-6 -9
	-6 -8
	1000 -1000

	-996 995 8 -10

Note

For the first test case, $\left(\frac{10}{1}, \frac{10}{1}\right) = (10, 10)$.

For the second test case, $\left(\frac{-1+5-4}{3}, \frac{-1-1+2}{3}\right) = (0, 0)$.

,

B. Minimize Equal Sum Subarrays

```
Input file:     standard input
Output file:    standard output
Time limit:     1.5 seconds
Memory limit:   256 megabytes
```

It is known that Farmer John likes Permutations, but I like them too!

— Sun Tzu, *The Art of Constructing Permutations*

You are given a permutation* p of length n.

Find a permutation q of length n that minimizes the number of pairs (i, j) $(1 \leq i \leq j \leq n)$ such that $p_i + p_{i+1} + \ldots + p_j = q_i + q_{i+1} + \ldots + q_j$.

*A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

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 n $(1 \leq n \leq 2 \cdot 10^5)$.

The following line contains n space-separated integers $p_1, p_2, \ldots, p_n$ $(1 \leq p_i \leq n)$ — denoting the permutation p of length n.

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

Output

For each test case, output one line containing any permutation of length n (the permutation q) such that q minimizes the number of pairs.

Standard Input	Standard Output

3
2
1 2
5
1 2 3 4 5
7
4 7 5 1 2 6 3
	2 1
	3 5 4 2 1
	6 2 1 4 7 3 5

Note

For the first test, there exists only one pair (i, j) $(1 \leq i \leq j \leq n)$ such that $p_i + p_{i+1} + \ldots + p_j = q_i + q_{i+1} + \ldots + q_j$, which is $(1, 2)$. It can be proven that no such q exists for which there are no pairs.

C. Perform Operations to Maximize Score

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

I see satyam343. I'm shaking. Please more median problems this time. I love those. Please satyam343 we believe in you.

— satyam343's biggest fan

You are given an array a of length n and an integer k. You are also given a binary array b of length n.

You can perform the following operation at most k times:

- Select an index i ($1 \leq i \leq n$) such that $b_i = 1$. Set $a_i = a_i + 1$ (i.e., increase a_i by 1).

Your *score* is defined to be $\max\limits_{i=1}^{n} \left(a_i + \operatorname{median}(c_i)\right)$, where c_i denotes the array of length $n-1$ that you get by deleting a_i from a. In other words, your score is the maximum value of $a_i + \operatorname{median}(c_i)$ over all i from 1 to n.

Find the maximum score that you can achieve if you perform the operations optimally.

For an arbitrary array p, $\operatorname{median}(p)$ is defined as the $\left\lfloor \frac{|p|+1}{2} \right\rfloor$-th **smallest** element of p. For example, $\operatorname{median}([3,2,1,3]) = 2$ and $\operatorname{median}([6,2,4,5,1]) = 4$.

Input

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

Each test case begins with two integers n and k ($2 \leq n \leq 2 \cdot 10^5$, $0 \leq k \leq 10^9$) — the length of the a and the number of operations you can perform.

The following line contains n space separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — denoting the array a.

The following line contains n space separated integers $b_1, b_2, \ldots, b_n$ (b_i is 0 or 1) — denoting the array b.

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

Output

For each test case, output the maximum value of score you can get on a new line.

Standard Input	Standard Output
8	16
2 10	6
3 3	8
1 1	13
3 10	21
3 3 3	26
0 0 0	

```
4 4
2 1 5 1
0 1 0 1
5 4
7 5 2 5 4
0 0 1 0 1
5 1
5 15 15 2 11
1 0 0 1 1
5 2
10 11 4 10 15
1 1 0 1 0
4 4
1 1 2 5
1 1 0 0
2 1000000000
1000000000 1000000000
1 1
```

```
8
3000000000
```

Note

For the first test case, it is optimal to perform 5 operations on both elements so $a = [8, 8]$. So, the maximum score we can achieve is $\max(8 + \mathrm{median}[8], 8 + \mathrm{median}[8]) = 16$, as $c_1 = [a_2] = [8]$. It can be proven that you cannot get a better score.

For the second test case, you are not able to perform operations on any elements, so a remains $[3, 3, 3]$. So, the maximum score we can achieve is $3 + \mathrm{median}[3, 3] = 6$, as $c_1 = [a_2, a_3] = [3, 3]$. It can be proven that you cannot get a better score.

,

D. Determine Winning Islands in Race

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

MOOOOOOOOOOOOOOOOO

— Bessie the Cow, *The Art of Racing on Islands*

Two of Farmer John's cows, Bessie and Elsie, are planning to race on n islands. There are $n-1$ *main* bridges, connecting island i to island $i+1$ for all $1 \leq i \leq n-1$. Additionally, there are m alternative bridges. Elsie can use **both** main and alternative bridges, while Bessie can only use main bridges. All bridges are **one way** and can only be used to travel from an island with a lower index to an island with a higher index.

Initially, Elsie starts on island 1, and Bessie starts on island s. The cows alternate turns, with Bessie making the first turn. Suppose the cow is on island i. During a cow's turn, if there are any bridges connecting island i to island j, then the cow can move to island j. Then, island i collapses, and all bridges connecting to island i also collapse. Also, note the following:

- If there are no bridges connecting island i to another island, then island i collapses, and this cow is eliminated from the race.
- If the other cow is also on island i, then after this cow moves to another island, the island collapses, and the other cow is eliminated from the race.
- After an island or bridge collapses, no cows may use them.
- If a cow is eliminated, their turn is skipped for the rest of the race.

The race ends once either cow reaches island n. It can be shown that regardless of the cows' strategies, at least one cow reaches island n. Bessie wins if and only if she reaches island n first.

For each $1 \leq s \leq n-1$, determine whether Bessie wins if she starts the race on island s. Assume both cows follow optimal strategies to ensure their own respective victories.

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 n and m ($2 \leq n \leq 2 \cdot 10^5$, $0 \leq m \leq 2 \cdot 10^5$) – the number of islands and the number of alternative bridges.

The next m lines of each test case contain u and v ($1 \leq u < v \leq n$) – the islands that the alternative bridge connects. It is guaranteed all alternative bridges are distinct, and they do not coincide with the main bridges.

It is guaranteed that neither the sum of n nor the sum of m over all test cases exceeds $2 \cdot 10^5$.

Output

For each test case, output a binary string of length $n-1$ on a new line. The i'th character is 1 if it is possible for Bessie to win if she starts on island i. Otherwise, it is 0.

Standard Input	Standard Output

5
6 0
6 1
2 6
	11111
	11011
	10011

6 1	100001111
1 5	11000111000111
10 4	
1 3	
1 6	
2 7	
3 8	
15 3	
2 8	
4 9	
8 15	

Note

In the first test case, there are no alternative bridges for Elsie to overtake Bessie and reach island n first, so Bessie will win on all islands because she always moves first.

In the second case, Bessie will lose if she starts on island 3 because:

- Bessie's Turn: Take a main bridge from island 3 to island 4.
- Elsie's Turn: Take a main bridge from island 1 to island 2.
- Bessie's Turn: Take a main bridge from island 4 to island 5.
- Elsie's Turn: Take an alternative bridge from island 2 to island 6. Elsie reaches island n first.

,

E1. Eliminating Balls With Merging (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

Drink water.

— Sun Tzu, *The Art of Becoming a Healthy Programmer*

This is the easy version of the problem. The only difference is that $x = n$ in this version. You must solve both versions to be able to hack.

You are given two integers n and x ($x = n$). There are n balls lined up in a row, numbered from 1 to n from left to right. Initially, there is a value a_i written on the i-th ball.

For each integer i from 1 to n, we define a function $f(i)$ as follows:

- Suppose you have a set $S = \{1, 2, \ldots, i\}$.
- In each operation, you have to select an integer l ($1 \le l < i$) from S such that l is not the largest element of S. Suppose r is the smallest element in S which is greater than l.
  - If $a_l > a_r$, you set $a_l = a_l + a_r$ and remove r from S.
  - If $a_l < a_r$, you set $a_r = a_l + a_r$ and remove l from S.
  - If $a_l = a_r$, you choose either the integer l or r to remove from S:
    - If you choose to remove l from S, you set $a_r = a_l + a_r$ and remove l from S.
    - If you choose to remove r from S, you set $a_l = a_l + a_r$ and remove r from S.
- $f(i)$ denotes the number of integers j ($1 \le j \le i$) such that it is possible to obtain $S = \{j\}$ after performing the above operations exactly $i - 1$ times.

For each integer i from x to n, you need to find $f(i)$.

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 two integers n and x ($1 \le n \le 2 \cdot 10^5; x = n$) — the number of balls and the smallest index i for which you need to find $f(i)$.

The second line of each test case contains $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the initial number written on each ball.

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

Output

For each test case, output $n - x + 1$ space separated integers on a new line, where the j-th integer should represent $f(x + j - 1)$.

Standard Input	Standard Output

3
5 5
1 2 3 2 1
7 7
	3
	4
	4

4 5 1 2 1 4 5	
11 11	
1 2 3 1 1 9 3 2 4 1 3	

Note

In the first test case, you are required to calculate $f(5)$. It can be shown that after 4 operations, S can contain 2, 3, or 4. The following shows the operations required to make $S = \{4\}$.

- Initially, $S = \{1, 2, 3, 4, 5\}$ and $a = [1, 2, 3, 2, 1]$.
- Choose $l = 1$. Naturally, $r = 2$. Since $a_1 < a_2$, we set $a_2 = 1 + 2$ and remove 1 from S. Now, $S = \{2, 3, 4, 5\}$ and $a = [1, 3, 3, 2, 1]$.
- Choose $l = 4$. Naturally, $r = 5$. Since $a_4 > a_5$, we set $a_4 = 2 + 1$ and remove 5 from S. Now, $S = \{2, 3, 4\}$ and $a = [1, 3, 3, 3, 1]$.
- Choose $l = 3$. Naturally, $r = 4$. Since $a_3 = a_4$, we have a choice whether to remove 3 or 4. Since we want to preserve 4, let's remove 3. So, set $a_4 = 3 + 3$ and remove 3 from S. Now, $S = \{2, 4\}$ and $a = [1, 3, 3, 6, 1]$.
- Choose $l = 2$. Naturally, $r = 4$. Since $a_2 < a_4$, we set $a_4 = 3 + 6$ and remove 2 from S. Finally, $S = \{4\}$ and $a = [1, 3, 3, 9, 1]$.

In the second test case, you are required to calculate $f(7)$. It can be shown that after 6 operations, S can contain 2, 4, 6, or 7.

,

E2. Eliminating Balls With Merging (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

Drink water.

— Sun Tzu, *The Art of Becoming a Healthy Programmer*

This is the hard version of the problem. The only difference is that $x = 1$ in this version. You must solve both versions to be able to hack.

You are given two integers n and x ($x = 1$). There are n balls lined up in a row, numbered from 1 to n from left to right. Initially, there is a value a_i written on the i-th ball.

For each integer i from 1 to n, we define a function $f(i)$ as follows:

- Suppose you have a set $S = \{1, 2, \ldots, i\}$.
- In each operation, you have to select an integer l ($1 \leq l < i$) from S such that l is not the largest element of S. Suppose r is the smallest element in S which is greater than l.
    - If $a_l > a_r$, you set $a_l = a_l + a_r$ and remove r from S.
    - If $a_l < a_r$, you set $a_r = a_l + a_r$ and remove l from S.
    - If $a_l = a_r$, you choose either the integer l or r to remove from S:
        - If you choose to remove l from S, you set $a_r = a_l + a_r$ and remove l from S.
        - If you choose to remove r from S, you set $a_l = a_l + a_r$ and remove r from S.
- $f(i)$ denotes the number of integers j ($1 \leq j \leq i$) such that it is possible to obtain $S = \{j\}$ after performing the above operations exactly $i - 1$ times.

For each integer i from x to n, you need to find $f(i)$.

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 x ($1 \leq n \leq 2 \cdot 10^5; x = 1$) — the number of balls and the smallest index i for which you need to find $f(i)$.

The second line of each test case contains $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the initial number written on each ball.

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

Output

For each test case, output $n - x + 1$ space separated integers on a new line, where the j-th integer should represent $f(x + j - 1)$.

Standard Input	Standard Output

3
5 1
1 2 3 2 1
7 1
	1 1 2 2 3
	1 1 1 1 1 3 4
	1 1 2 2 2 1 1 1 3 3 4

4 5 1 2 1 4 5	
11 1	
1 2 3 1 1 9 3 2 4 1 3	

Note

In the first test case, below are the possible values of j for each $f(i)$ from 1 to n.

- For $f(1)$, the only possible value of j is 1.
- For $f(2)$, the only possible value of j is 2.
- For $f(3)$, the possible values of j are 2 and 3.
- For $f(4)$, the possible values of j are 2 and 3.
- For $f(5)$, the possible values of j are 2, 3, and 4.