EPS 2024-02-05

A. Pursuit

2 seconds, 512 megabytes

You and your friend Ilya are participating in an individual programming contest consisting of multiple stages. A contestant can get between 0 and 100 points, inclusive, for each stage, independently of other contestants.

Points received by contestants in different stages are used for forming overall contest results. Suppose that k stages of the contest are completed. For each contestant, $k - \lfloor \frac{k}{4} \rfloor$ stages with the highest scores are selected, and these scores are added up. This sum is the overall result of the contestant. (Here $\lfloor t \rfloor$ denotes rounding t down.)

For example, suppose 9 stages are completed, and your scores are $50, 30, 50, 50, 100, 10, 30, 100, 50$. First, 7 stages with the highest scores are chosen — for example, all stages except for the 2-nd and the 6-th can be chosen. Then your overall result is equal to $50 + 50 + 50 + 100 + 30 + 100 + 50 = 430$.

As of now, n stages are completed, and you know the points you and Ilya got for these stages. However, it is unknown how many more stages will be held. You wonder what the smallest number of additional stages is, after which your result might become greater than or equal to Ilya's result, at least in theory. Find this number!

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of completed stages.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 100$) — your points for the completed stages.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 100$) — Ilya's points for the completed stages.

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

Output

For each test case print a single integer — the smallest number of additional stages required for your result to be able to become greater than or equal to Ilya's result.

If your result is already not less than Ilya's result, print 0.

input

```
5
1
100
0
1
0
100
4
20 30 40 50
100 100 100 100
4
10 20 30 40
100 100 100 100
7
7 59 62 52 27 31 55
33 35 50 98 83 80 64
```

output

```
0
1
3
4
2
```

In the first test case, you have scored 100 points for the first stage, while Ilya has scored 0. Thus, your overall result (100) is already not less than Ilya's result (0).

In the second test case, you have scored 0 points for the first stage, while Ilya has scored 100. A single stage with an opposite result is enough for both your and Ilya's overall scores to become equal to 100.

In the third test case, your overall result is $30 + 40 + 50 = 120$, while Ilya's result is $100 + 100 + 100 = 300$. After three additional stages your result might become equal to 420, while Ilya's result might become equal to 400.

In the fourth test case, your overall result after four additional stages might become equal to 470, while Ilya's result might become equal to 400. Three stages are not enough.

B. Koxia and Whiteboards

1 second, 256 megabytes

Kiyora has n whiteboards numbered from 1 to n. Initially, the i-th whiteboard has the integer a_i written on it.

Koxia performs m operations. The j-th operation is to choose one of the whiteboards and change the integer written on it to b_j.

Find the maximum possible sum of integers written on the whiteboards after performing all m operations.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. The description of test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 100$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

The third line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^9$).

Output

For each test case, output a single integer — the maximum possible sum of integers written on whiteboards after performing all m operations.

input

```
4
3 2
1 2 3
4 5
2 3
1 2
3 4 5
1 1
100
1
5 3
1 1 1 1 1
1000000000 1000000000 1000000000
```

output

```
12
9
1
3000000002
```

In the first test case, Koxia can perform the operations as follows:

1. Choose the 1-st whiteboard and rewrite the integer written on it to $b_1 = 4$.
2. Choose the 2-nd whiteboard and rewrite to $b_2 = 5$.

After performing all operations, the numbers on the three whiteboards are 4, 5 and 3 respectively, and their sum is 12. It can be proven that this is the maximum possible sum achievable.

In the second test case, Koxia can perform the operations as follows:

1. Choose the 2-nd whiteboard and rewrite to $b_1 = 3$.
2. Choose the 1-st whiteboard and rewrite to $b_2 = 4$.
3. Choose the 2-nd whiteboard and rewrite to $b_3 = 5$.

The sum is $4 + 5 = 9$. It can be proven that this is the maximum possible sum achievable.

C. Rocket

1 second, 256 megabytes

This is an interactive problem.

Natasha is going to fly to Mars. Finally, Natasha sat in the rocket. She flies, flies... but gets bored. She wishes to arrive to Mars already! So she decides to find something to occupy herself. She couldn't think of anything better to do than to calculate the distance to the red planet.

Let's define x as the distance to Mars. Unfortunately, Natasha does not know x. But it is known that $1 \le x \le m$, where Natasha knows the number m. Besides, x and m are positive integers.

Natasha can ask the rocket questions. Every question is an integer y ($1 \le y \le m$). The correct answer to the question is -1, if $x < y$, 0, if $x = y$, and 1, if $x > y$. But the rocket is broken — it does not always answer correctly. Precisely: let the correct answer to the current question be equal to t, then, if the rocket answers this question correctly, then it will answer t, otherwise it will answer $-t$.

In addition, the rocket has a sequence p of length n. Each element of the sequence is either 0 or 1. The rocket processes this sequence in the cyclic order, that is 1-st element, 2-nd, 3-rd, $\ldots$, $(n-1)$-th, n-th, 1-st, 2-nd, 3-rd, $\ldots$, $(n-1)$-th, n-th, $\ldots$. If the current element is 1, the rocket answers correctly, if 0 — lies. Natasha doesn't know the sequence p, but she knows its length — n.

You can ask the rocket no more than 60 questions.

Help Natasha find the distance to Mars. Assume, that the distance to Mars does not change while Natasha is asking questions.

Your solution will not be accepted, if it does not receive an answer 0 from the rocket (even if the distance to Mars is uniquely determined by the already received rocket's answers).

Input

The first line contains two integers m and n ($1 \le m \le 10^9$, $1 \le n \le 30$) — the maximum distance to Mars and the number of elements in the sequence p.

Interaction

You can ask the rocket no more than 60 questions.

To ask a question, print a number y ($1 \le y \le m$) and an end-of-line character, then do the operation `flush` and read the answer to the question.

If the program reads 0, then the distance is correct and you must immediately terminate the program (for example, by calling `exit(0)`). If you ignore this, you can get any verdict, since your program will continue to read from the closed input stream.

If at some point your program reads -2 as an answer, it must immediately end (for example, by calling `exit(0)`). You will receive the "Wrong answer" verdict, and this will mean that the request is incorrect or the number of requests exceeds 60. If you ignore this, you can get any verdict, since your program will continue to read from the closed input stream.

If your program's request is not a valid integer between -2^{31} and $2^{31} - 1$ (inclusive) without leading zeros, then you can get any verdict.

You can get "Idleness limit exceeded" if you don't print anything or if you forget to flush the output.

To flush the output buffer you can use (after printing a query and end-of-line):

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

Hacking

Use the following format for hacking:

In the first line, print 3 integers m, n, x ($1 \le x \le m \le 10^9$, $1 \le n \le 30$) — the maximum distance to Mars, the number of elements in the sequence p and the current distance to Mars.

In the second line, enter n numbers, each of which is equal to 0 or 1 — sequence p.

The hacked solution will not have access to the number x and sequence p.

input

```
5 2
1
-1
-1
1
0
```

output

```
1
2
4
5
3
```

In the example, hacking would look like this:

```
5 2 3

1 0
```

This means that the current distance to Mars is equal to 3, Natasha knows that it does not exceed 5, and the rocket answers in order: correctly, incorrectly, correctly, incorrectly ...

Really:

on the first query (1) the correct answer is 1, the rocket answered correctly: 1;

on the second query (2) the correct answer is 1, the rocket answered incorrectly: -1;

on the third query (4) the correct answer is -1, the rocket answered correctly: -1;

on the fourth query (5) the correct answer is -1, the rocket answered incorrectly: 1;

on the fifth query (3) the correct and incorrect answer is 0.

D. Flexible String

4.0 s, 256 megabytes

You have a string a and a string b. Both of the strings have length n. There are **at most 10 different characters** in the string a. You also have a set Q. Initially, the set Q is empty. You can apply the following operation on the string a any number of times:

- Choose an index i $(1 \le i \le n)$ and a lowercase English letter c. Add a_i to the set Q and then replace a_i with c.

For example, Let the string a be "`abecca`". We can do the following operations:

- In the first operation, if you choose $i = 3$ and $c = \texttt{x}$, the character $a_3 = \texttt{e}$ will be added to the set Q. So, the set Q will be $\{\texttt{e}\}$, and the string a will be "`abxcca`".
- In the second operation, if you choose $i = 6$ and $c = \texttt{s}$, the character $a_6 = \texttt{a}$ will be added to the set Q. So, the set Q will be $\{\texttt{e, a}\}$, and the string a will be "`abxccs`".

You can apply any number of operations on a, but in the end, the set Q should contain **at most k different characters**. Under this constraint, you have to maximize the number of integer pairs (l, r) $(1 \le l \le r \le n)$ such that $a[l, r] = b[l, r]$. Here, $s[l, r]$ means the substring of string s starting at index l (inclusively) and ending at index r (inclusively).

Input

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

The first line contains two integers n and k $(1 \le n \le 10^5$, $0 \le k \le 10)$ — the length of the two strings and the limit on different characters in the set Q.

The second line contains the string a of length n. There is **at most 10 different characters** in the string a.

The last line contains the string b of length n.

Both of the strings a and b contain only lowercase English letters. The sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, print a single integer in a line, the maximum number of pairs (l, r) satisfying the constraints.

input

```
6
3 1
abc
abd
3 0
abc
abd
3 1
xbb
xcd
4 1
abcd
axcb
3 10
abc
abd
10 3
lkwhbahuqa
qoiujoncjb
```

output

```
6
3
6
6
6
11
```

In the first case, we can select index $i = 3$ and replace it with character $c = \texttt{d}$. All possible pairs (l, r) will be valid.

In the second case, we can't perform any operation. The 3 valid pairs (l, r) are:

1. $a[1, 1] = b[1, 1] =$ "`a`",
2. $a[1, 2] = b[1, 2] =$ "`ab`",
3. $a[2, 2] = b[2, 2] =$ "`b`".

In the third case, we can choose index 2 and index 3 and replace them with the characters `c` and `d` respectively. The final set Q will be $\{\texttt{b}\}$ having size 1 that satisfies the value of k. All possible pairs (l, r) will be valid.

E_Bonus. Teleporters (Hard Version)

1 second, 256 megabytes

The only difference between the easy and hard versions are the locations you can teleport to.

Consider the points $0, 1, \dots, n+1$ on the number line. There is a teleporter located on each of the points $1, 2, \dots, n$. At point i, you can do the following:

- Move left one unit: it costs 1 coin.
- Move right one unit: it costs 1 coin.
- Use a teleporter at point i, if it exists: it costs a_i coins. As a result, you can choose whether to teleport to point 0 or point $n+1$. Once you use a teleporter, you **can't** use it again.

You have c coins, and you start at point 0. What's the most number of teleporters you can use?

Input

The input consists of multiple test cases. The first line contains an integer t $(1 \le t \le 1000)$ — the number of test cases. The descriptions of the test cases follow.

The first line of each test case contains two integers n and c ($1 \le n \le 2 \cdot 10^5; 1 \le c \le 10^9$) — the length of the array and the number of coins you have respectively.

The following line contains n space-separated positive integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the costs to use the teleporters.

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 number of teleporters you can use.

input

```
10
5 6
1 1 1 1 1
8 32
100 52 13 6 9 4 100 35
1 1
5
4 5
4 3 2 1
5 9
2 3 1 4 1
5 8
2 3 1 4 1
4 3
2 3 4 1
4 9
5 4 3 3
2 14
7 5
5 600000000
500000000 400000000 300000000 200000000 100000000
```

output

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

In the first test case, you can move one unit to the right, use the teleporter at index 1 and teleport to point $n+1$, move one unit to the left and use the teleporter at index 5. You are left with $6-1-1-1-1=2$ coins, and wherever you teleport, you won't have enough coins to use another teleporter. You have used two teleporters, so the answer is two.

In the second test case, you go four units to the right and use the teleporter to go to $n+1$, then go three units left and use the teleporter at index 6 to go to $n+1$, and finally, you go left four times and use the teleporter. The total cost will be $4+6+3+4+4+9=30$, and you used three teleporters.

In the third test case, you don't have enough coins to use any teleporter, so the answer is zero.

F_Bonus. Remove the Bracket

1 second, 256 megabytes

RSJ has a sequence a of n integers $a_1, a_2, \ldots, a_n$ and an integer s. For each of $a_2, a_3, \ldots, a_{n-1}$, he chose a pair of **non-negative integers** x_i and y_i such that $x_i + y_i = a_i$ and $(x_i - s) \cdot (y_i - s) \ge 0$.

Now he is interested in the value

$$F = a_1 \cdot x_2 + y_2 \cdot x_3 + y_3 \cdot x_4 + \ldots + y_{n-2} \cdot x_{n-1} + y_{n-1} \cdot a_n.$$

Please help him find the minimum possible value F he can get by choosing x_i and y_i optimally. It can be shown that there is always at least one valid way to choose them.

Input

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

The first line of each test case contains two integers n, s ($3 \le n \le 2 \cdot 10^5; 0 \le s \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 2 \cdot 10^5)$.

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

Output

For each test case, print the minimum possible value of F.

input

```
10
5 0
2 0 1 3 4
5 1
5 3 4 3 5
7 2
7 6 5 4 3 2 1
5 1
1 2 3 4 5
5 2
1 2 3 4 5
4 0
0 1 1 1
5 5
4 3 5 6 4
4 1
0 2 1 0
3 99999
200000 200000 200000
6 8139
7976 129785 12984 78561 173685 15480
```

output

```
0
18
32
11
14
0
16
0
40000000000
2700826806
```

In the first test case, $2 \cdot 0 + 0 \cdot 1 + 0 \cdot 3 + 0 \cdot 4 = 0$.

In the second test case, $5 \cdot 1 + 2 \cdot 2 + 2 \cdot 2 + 1 \cdot 5 = 18$.

G_Bonus. The Parade

2 seconds, 512 megabytes

The Berland Army is preparing for a large military parade. It is already decided that the soldiers participating in it will be divided into k rows, and all rows will contain *the same* number of soldiers.

Of course, not every arrangement of soldiers into k rows is suitable. Heights of all soldiers in the same row should not differ by more than 1. The height of each soldier is an integer between 1 and n.

For each possible height, you know the number of soldiers having this height. To conduct a parade, you have to choose the soldiers participating in it, and then arrange *all of the chosen soldiers* into k rows so that both of the following conditions are met:

- each row has the same number of soldiers,
- no row contains a pair of soldiers such that their heights differ by 2 or more.

Calculate the maximum number of soldiers who can participate in the parade.

Input

The first line contains one integer t ($1 \le t \le 10000$) — the number of test cases. Then the test cases follow.

Each test case begins with a line containing two integers n and k ($1 \le n \le 30000$, $1 \le k \le 10^{12}$) — the number of different heights of soldiers and the number of rows of soldiers in the parade, respectively.

The second (and final) line of each test case contains n integers c_1 , c_2 , ..., c_n ($0 \le c_i \le 10^{12}$), where c_i is the number of soldiers having height i in the Berland Army.

It is guaranteed that the sum of n over all test cases does not exceed 30000.

Output

For each test case, print one integer — the maximum number of soldiers that can participate in the parade.

input

```
5
3 4
7 1 13
1 1
100
1 3
100
2 1
1000000000000 1000000000000
4 1
10 2 11 1
```

output

```
16
100
99
2000000000000
13
```

Explanations for the example test cases:

1. the heights of soldiers in the rows can be: $[3, 3, 3, 3]$, $[1, 2, 1, 1]$, $[1, 1, 1, 1]$, $[3, 3, 3, 3]$ (each list represents a row);
2. all soldiers can march in the same row;
3. 33 soldiers with height 1 in each of 3 rows;
4. all soldiers can march in the same row;
5. all soldiers with height 2 and 3 can march in the same row.