# Codeforces Round #590 (Div. 3)

## A. Equalize Prices Again

1 second, 256 megabytes

You are both a shop keeper and a shop assistant at a small nearby shop. You have $n$ goods, the $i$-th good costs $a_i$ coins.

You got tired of remembering the price of each product when customers ask for it, thus you decided to simplify your life. More precisely you decided to set the same price for all $n$ goods you have.

However, you don't want to lose any money so you want to choose the price in such a way that the sum of new prices is not less than the sum of the initial prices. It means that if you sell all $n$ goods for the new price, you will receive at least the same (or greater) amount of money as if you sell them for their initial prices.

On the other hand, you don't want to lose customers because of big prices so among all prices you can choose you need to choose the minimum one.

So you need to find the minimum possible equal price of all $n$ goods so if you sell them for this price, you will receive at least the same (or greater) amount of money as if you sell them for their initial prices.

You have to answer $q$ independent queries.

### Input

The first line of the input contains one integer $q$ $(1 \le q \le 100)$ — the number of queries. Then $q$ queries follow.

The first line of the query contains one integer $n$ $(1 \le n \le 100)$ — the number of goods. The second line of the query contains $n$ integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^7)$, where $a_i$ is the price of the $i$-th good.

### Output

For each query, print the answer for it — the minimum possible equal price of all $n$ goods so if you sell them for this price, you will receive at least the same (or greater) amount of money as if you sell them for their initial prices.

input

```
3
5
1 2 3 4 5
3
1 2 2
4
1 1 1 1
```

output

```
3
2
1
```

## B1. Social Network (easy version)

2 seconds, 256 megabytes

**The only difference between easy and hard versions are constraints on $n$ and $k$.**

You are messaging in one of the popular social networks via your smartphone. Your smartphone can show at most $k$ most recent conversations with your friends. Initially, the screen is empty (i.e. the number of displayed conversations equals $0$).

Each conversation is between you and some of your friends. There is at most one conversation with any of your friends. So each conversation is uniquely defined by your friend.

You (suddenly!) have the ability to see the future. You know that during the day you will receive $n$ messages, the $i$-th message will be received from the friend with ID $id_i$ ($1 \le id_i \le 10^9$).

If you receive a message from $id_i$ in the conversation which is currently displayed on the smartphone then nothing happens: the conversations of the screen do not change and do not change their order, you read the message and continue waiting for new messages.

Otherwise (i.e. if there is no conversation with $id_i$ on the screen):

- Firstly, if the number of conversations displayed on the screen is $k$, the last conversation (which has the position $k$) is removed from the screen.
- Now the number of conversations on the screen is guaranteed to be less than $k$ and the conversation with the friend $id_i$ is not displayed on the screen.
- The conversation with the friend $id_i$ appears on the first (the topmost) position on the screen and all the other displayed conversations are shifted one position down.

Your task is to find the list of conversations (in the order they are displayed on the screen) after processing all $n$ messages.

## Input

The first line of the input contains two integers $n$ and $k$ $(1 \le n, k \le 200)$ — the number of messages and the number of conversations your smartphone can show.

The second line of the input contains $n$ integers $id_1, id_2, \dots, id_n$ $(1 \le id_i \le 10^9)$, where $id_i$ is the ID of the friend which sends you the $i$-th message.

## Output

In the first line of the output print one integer $m$ $(1 \le m \le min(n, k))$ — the number of conversations shown after receiving all $n$ messages.

In the second line print $m$ integers $ids_1, ids_2, \dots, ids_m$, where $ids_i$ should be equal to the ID of the friend corresponding to the conversation displayed on the position $i$ after receiving all $n$ messages.

input
```
7 2
1 2 3 2 1 3 2
```
output
```
2
2 1
```

input
```
10 4
2 3 3 1 1 2 1 2 3 3
```
output
```
3
1 3 2
```

In the first example the list of conversations will change in the following way (in order from the first to last message):

- $[]$;
- $[1]$;
- $[2, 1]$;
- $[3, 2]$;
- $[3, 2]$;
- $[1, 3]$;
- $[1, 3]$;
- $[2, 1]$.

In the second example the list of conversations will change in the following way:

- $[]$;
- $[2]$;
- $[3, 2]$;
- $[3, 2]$;
- $[1, 3, 2]$;
- and then the list will not change till the end.

## B2. Social Network (hard version)

2 seconds, 256 megabytes

**The only difference between easy and hard versions are constraints on $n$ and $k$.**

You are messaging in one of the popular social networks via your smartphone. Your smartphone can show at most $k$ most recent conversations with your friends. Initially, the screen is empty (i.e. the number of displayed conversations equals $0$).

Each conversation is between you and some of your friends. There is at most one conversation with any of your friends. So each conversation is uniquely defined by your friend.

You (suddenly!) have the ability to see the future. You know that during the day you will receive $n$ messages, the $i$-th message will be received from the friend with ID $id_i$ ($1 \le id_i \le 10^9$).

If you receive a message from $id_i$ in the conversation which is currently displayed on the smartphone then nothing happens: the conversations of the screen do not change and do not change their order, you read the message and continue waiting for new messages.

Otherwise (i.e. if there is no conversation with $id_i$ on the screen):

- Firstly, if the number of conversations displayed on the screen is $k$, the last conversation (which has the position $k$) is removed from the screen.
- Now the number of conversations on the screen is guaranteed to be less than $k$ and the conversation with the friend $id_i$ is not displayed on the screen.
- The conversation with the friend $id_i$ appears on the first (the topmost) position on the screen and all the other displayed conversations are shifted one position down.

Your task is to find the list of conversations (in the order they are displayed on the screen) after processing all $n$ messages.

### Input

The first line of the input contains two integers $n$ and $k$ ($1 \le n, k \le 2 \cdot 10^5$) — the number of messages and the number of conversations your smartphone can show.

The second line of the input contains $n$ integers $id_1, id_2, \dots, id_n$ ($1 \le id_i \le 10^9$), where $id_i$ is the ID of the friend which sends you the $i$-th message.

### Output

In the first line of the output print one integer $m$ ($1 \le m \le min(n, k)$) — the number of conversations shown after receiving all $n$ messages.

In the second line print $m$ integers $ids_1, ids_2, \dots, ids_m$, where $ids_i$ should be equal to the ID of the friend corresponding to the conversation displayed on the position $i$ after receiving all $n$ messages.

input

```
7 2
1 2 3 2 1 3 2
```

output

```
2
2 1
```

input

```
10 4
2 3 3 1 1 2 1 2 3 3
```

output

```
3
1 3 2
```

In the first example the list of conversations will change in the following way (in order from the first to last message):

- $[]$;
- $[1]$;
- $[2, 1]$;
- $[3, 2]$;
- $[3, 2]$;
- $[1, 3]$;
- $[1, 3]$;
- $[2, 1]$.

In the second example the list of conversations will change in the following way:

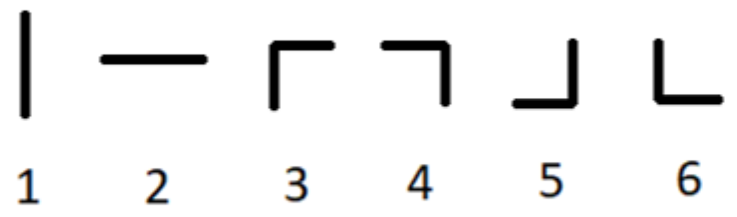
- $[]$;
- $[2]$;
- $[3, 2]$;
- $[3, 2]$;
- $[1, 3, 2]$;
- and then the list will not change till the end.

## C. Pipes

2 seconds, 256 megabytes

You are given a system of pipes. It consists of two rows, each row consists of $n$ pipes. The top left pipe has the coordinates $(1, 1)$ and the bottom right — $(2, n)$.

There are six types of pipes: two types of straight pipes and four types of curved pipes. Here are the examples of all six types:

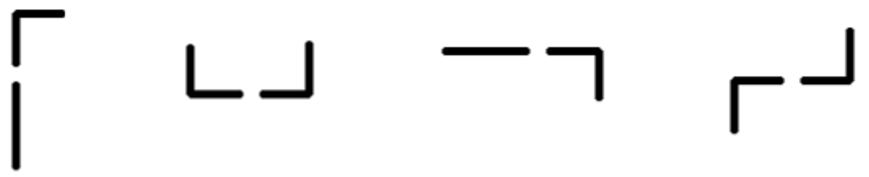


Types of pipes

You can turn each of the given pipes $90$ degrees clockwise or counterclockwise **arbitrary (possibly, zero) number of times** (so the types $1$ and $2$ can become each other and types $3, 4, 5, 6$ can become each other).

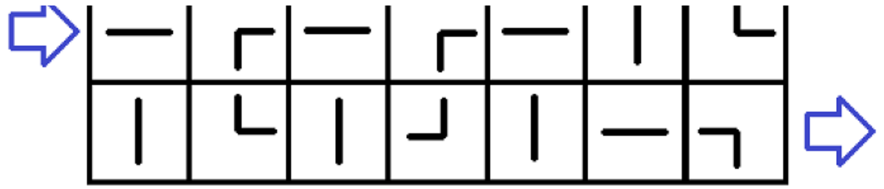
You want to turn some pipes in a way that the water flow can start at $(1, 0)$ (to the left of the top left pipe), move to the pipe at $(1, 1)$, flow somehow by **connected pipes** to the pipe at $(2, n)$ and flow right to $(2, n + 1)$.

Pipes are connected if they are adjacent in the system and their ends are connected. Here are examples of connected pipes:



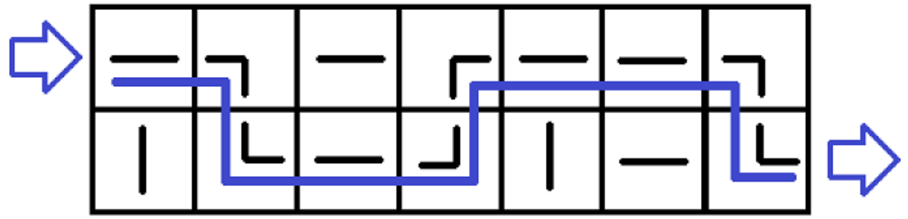
Examples of connected pipes

Let's describe the problem using some example:



The first example input

And its solution is below:



The first example answer

As you can see, the water flow is the poorly drawn blue line. To obtain the answer, we need to turn the pipe at $(1, 2)$ $90$ degrees clockwise, the pipe at $(2, 3)$ $90$ degrees, the pipe at $(1, 6)$ $90$ degrees, the pipe at $(1, 7)$ $180$ degrees and the pipe at $(2, 7)$ $180$ degrees. Then the flow of water can reach $(2, n + 1)$ from $(1, 0)$.

You have to answer $q$ independent queries.

## Input

The first line of the input contains one integer $q$ ($1 \le q \le 10^4$) — the number of queries. Then $q$ queries follow.

Each query consists of exactly three lines. The first line of the query contains one integer $n$ ($1 \le n \le 2 \cdot 10^5$) — the number of pipes in each row. The next two lines contain a description of the first and the second rows correspondingly. Each row description consists of $n$ digits from $1$ to $6$ without any whitespaces between them, each digit corresponds to the type of pipe in the corresponding cell. See the problem statement to understand which digits correspond to which types of pipes.

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

### Output

For the $i$-th query print the answer for it — "`YES`" (without quotes) if it is possible to turn some pipes in a way that the water flow can reach $(2, n+1)$ from $(1, 0)$, and "`NO`" otherwise.

input

```
6
7
2323216
1615124
1
3
4
2
13
24
2
12
34
3
536
345
2
46
54
```

output

```
YES
YES
YES
NO
YES
NO
```

The first query from the example is described in the problem statement.

## D. Distinct Characters Queries

2 seconds, 256 megabytes

You are given a string $s$ consisting of lowercase Latin letters and $q$ queries for this string.

Recall that the substring $s[l; r]$ of the string $s$ is the string $s_l s_{l+1} \dots s_r$. For example, the substrings of "`codeforces`" are "`code`", "`force`", "`f`", "`for`", but not "`coder`" and "`top`".

There are two types of queries:

- $1\ pos\ c$ ($1 \le pos \le |s|$, $c$ is lowercase Latin letter): replace $s_{pos}$ with $c$ (set $s_{pos} := c$);
- $2\ l\ r$ ($1 \le l \le r \le |s|$): calculate the number of distinct characters in the substring $s[l; r]$.

**Input**

The first line of the input contains one string $s$ consisting of no more than $10^5$ lowercase Latin letters.

The second line of the input contains one integer $q$ ($1 \le q \le 10^5$) — the number of queries.

The next $q$ lines contain queries, one per line. Each query is given in the format described in the problem statement. It is guaranteed that there is at least one query of the second type.

**Output**

For each query of the second type print the answer for it — the number of distinct characters in the required substring in this query.

input

```
abacaba
5
2 1 4
1 4 b
1 5 b
2 4 6
2 1 7
```

output

```
3
1
2
```

input

```
dfcbbcfeeedbaea
15
1 6 e
1 4 b
2 6 14
1 7 b
1 12 c
2 6 8
2 1 6
1 7 c
1 2 f
1 10 a
2 7 9
1 10 a
1 14 b
1 1 f
2 1 11
```

output

```
5
2
5
2
6
```

## E. Special Permutations

2 seconds, 256 megabytes

Let's define $p_i(n)$ as the following permutation: $[i, 1, 2, \dots, i-1, i+1, \dots, n]$. This means that the $i$-th permutation is **almost identity** (i.e. which maps every element to itself) permutation but the element $i$ is on the first position. Examples:

- $p_1(4) = [1, 2, 3, 4]$;
- $p_2(4) = [2, 1, 3, 4]$;
- $p_3(4) = [3, 1, 2, 4]$;
- $p_4(4) = [4, 1, 2, 3]$.

You are given an array $x_1, x_2, \dots, x_m$ $(1 \le x_i \le n)$.

Let $pos(p, val)$ be the position of the element $val$ in $p$. So, $pos(p_1(4), 3) = 3, pos(p_2(4), 2) = 1, pos(p_4(4), 4) = 1$.

Let's define a function $f(p) = \sum\limits_{i=1}^{m-1} |pos(p, x_i) - pos(p, x_{i+1})|$, where $|val|$ is the absolute value of $val$. This function means the sum of distances between adjacent elements of $x$ in $p$.

Your task is to calculate $f(p_1(n)), f(p_2(n)), \dots, f(p_n(n))$.

### Input

The first line of the input contains two integers $n$ and $m$ $(2 \le n, m \le 2 \cdot 10^5)$ — the number of elements in each permutation and the number of elements in $x$.

The second line of the input contains $m$ integers ($m$, **not** $n$) $x_1, x_2, \dots, x_m$ $(1 \le x_i \le n)$, where $x_i$ is the $i$-th element of $x$. Elements of $x$ can repeat and appear in arbitrary order.

### Output

Print $n$ integers: $f(p_1(n)), f(p_2(n)), \dots, f(p_n(n))$.

input

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

output

```
3 4 6 5
```

input

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

output

```
9 8 12 6 8
```

input

```
2 10
1 2 1 1 2 2 2 2 2 2
```

output

```
3 3
```

Consider the first example:

$x = [1, 2, 3, 4]$, so

- for the permutation $p_1(4) = [1, 2, 3, 4]$ the answer is $|1 - 2| + |2 - 3| + |3 - 4| = 3$;
- for the permutation $p_2(4) = [2, 1, 3, 4]$ the answer is $|2 - 1| + |1 - 3| + |3 - 4| = 4$;
- for the permutation $p_3(4) = [3, 1, 2, 4]$ the answer is $|2 - 3| + |3 - 1| + |1 - 4| = 6$;
- for the permutation $p_4(4) = [4, 1, 2, 3]$ the answer is $|2 - 3| + |3 - 4| + |4 - 1| = 5$.

Consider the second example:

$x = [2, 1, 5, 3, 5]$, so

- for the permutation $p_1(5) = [1, 2, 3, 4, 5]$ the answer is $|2 - 1| + |1 - 5| + |5 - 3| + |3 - 5| = 9$;
- for the permutation $p_2(5) = [2, 1, 3, 4, 5]$ the answer is $|1 - 2| + |2 - 5| + |5 - 3| + |3 - 5| = 8$;
- for the permutation $p_3(5) = [3, 1, 2, 4, 5]$ the answer is $|3 - 2| + |2 - 5| + |5 - 1| + |1 - 5| = 12$;
- for the permutation $p_4(5) = [4, 1, 2, 3, 5]$ the answer is $|3 - 2| + |2 - 5| + |5 - 4| + |4 - 5| = 6$;
- for the permutation $p_5(5) = [5, 1, 2, 3, 4]$ the answer is $|3 - 2| + |2 - 1| + |1 - 4| + |4 - 1| = 8$.

## F. Yet Another Substring Reverse

2 seconds, 256 megabytes

You are given a string $s$ consisting only of first $20$ lowercase Latin letters ('a', 'b', ..., 't').

Recall that the substring $s[l; r]$ of the string $s$ is the string $s_l s_{l+1} \dots s_r$. For example, the substrings of "codeforces" are "code", "force", "f", "for", but not "coder" and "top".

You can perform the following operation **no more than once**: choose some substring $s[l; r]$ and **reverse** it (i.e. the string $s_l s_{l+1} \dots s_r$ becomes $s_r s_{r-1} \dots s_l$).

Your goal is to maximize the length of the maximum substring of $s$ consisting of **distinct** (i.e. unique) characters.

The string consists of **distinct** characters if no character in this string appears more than once. For example, strings "abcde", "arctg" and "minecraft" consist of distinct characters but strings "codeforces", "abacaba" do not consist of distinct characters.

### Input

The only line of the input contains one string $s$ consisting of no more than $10^6$ characters 'a', 'b', ..., 't' (first $20$ lowercase Latin letters).

### Output

Print one integer — the maximum possible length of the maximum substring of $s$ consisting of distinct characters after reversing no more than one its substring.

input
```
abacaba
```
output
```
3
```

input
```
aabbcc
```
output
```
3
```

input
```
abcdecdf
```
output
```
6
```

input
```
abcdeefc
```
output
```
6
```