Codeforces Round #741 (Div. 2)

A. The Miracle and the Sleeper

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integers l and r, $l \le r$. Find the largest possible value of $a \bmod b$ over all pairs (a, b) of integers for which $r \ge a \ge b \ge l$.

As a reminder, $a \bmod b$ is a remainder we get when dividing a by b. For example, $26 \bmod 8 = 2$.

Input
Each test contains multiple test cases.

The first line contains one positive integer t $(1 \le t \le 10^4)$, denoting the number of test cases. Description of the test cases follows.

The only line of each test case contains two integers l, r $(1 \le l \le r \le 10^9)$.

Output
For every test case, output the largest possible value of $a \bmod b$ over all pairs (a, b) of integers for which $r \ge a \ge b \ge l$.

Example

input
4 1 1 999999999 1000000000 8 26 1 999999999
output
0 1 12 499999999

Note
In the first test case, the only allowed pair is $(a, b) = (1, 1)$, for which $a \bmod b = 1 \bmod 1 = 0$.

In the second test case, the optimal choice is pair $(a, b) = (1000000000, 999999999)$, for which $a \bmod b = 1$.

B. Scenes From a Memory

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

During the hypnosis session, Nicholas suddenly remembered a positive integer n, which **doesn't contain zeros in decimal notation**.

Soon, when he returned home, he got curious: what is the maximum number of digits that can be removed from the number so that the number becomes **not prime**, that is, either composite or equal to one?

For some numbers doing so is impossible: for example, for number 53 it's impossible to delete some of its digits to obtain a not prime integer. However, **for all n in the test cases of this problem, it's guaranteed that it's possible to delete some of their digits to obtain a not prime number**.

Note that you cannot remove all the digits from the number.

A prime number is a number that has no divisors except one and itself. A composite is a number that has more than two divisors. 1 is neither a prime nor a composite number.

Input
Each test contains multiple test cases.

The first line contains one positive integer t $(1 \le t \le 10^3)$, denoting the number of test cases. Description of the test cases follows.

The first line of each test case contains one positive integer k $(1 \le k \le 50)$ — the number of digits in the number.

The second line of each test case contains a positive integer n, which doesn't contain zeros in decimal notation ($10^{k-1} \le n < 10^k$). It is guaranteed that it is always possible to remove less than k digits to make the number not prime.

It is guaranteed that the sum of k over all test cases does not exceed 10^4.

Output

For every test case, print two numbers in two lines. In the first line print the number of digits, that you **have left** in the number. In the second line print the digits left after all delitions.

If there are multiple solutions, print any.

Example

input
7 3 237 5 44444 3 221 2 35 3 773 1 4 30 626221626221626221626221626221
output
2 27 1 4 1 1 2 35 2 77 1 4 1 6

Note

In the first test case, you can't delete 2 digits from the number 237, as all the numbers 2, 3, and 7 are prime. However, you can delete 1 digit, obtaining a number $27 = 3^3$.

In the second test case, you can delete all digits except one, as $4 = 2^2$ is a composite number.

C. Rings

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

> Frodo was caught by Saruman. He tore a pouch from Frodo's neck, shook out its contents —there was a pile of different rings: gold and silver... "How am I to tell which is the One?!" the mage howled.
>
> "Throw them one by one into the Cracks of Doom and watch when Mordor falls!"

Somewhere in a parallel Middle-earth, when Saruman caught Frodo, he only found n rings. And the i-th ring was either gold or silver. For convenience Saruman wrote down a binary string s of n characters, where the i-th character was 0 if the i-th ring was gold, and 1 if it was silver.

Saruman has a magic function f, which takes a binary string and returns a number obtained by converting the string into a binary number and then converting the binary number into a decimal number. For example,
$f(001010) = 10, f(111) = 7, f(11011101) = 221$.

Saruman, however, thinks that the order of the rings plays some important role. He wants to find 2 pairs of integers $(l_1, r_1), (l_2, r_2)$, such that:

- $1 \le l_1 \le n$, $1 \le r_1 \le n$, $r_1 - l_1 + 1 \ge \lfloor \frac{n}{2} \rfloor$
- $1 \le l_2 \le n$, $1 \le r_2 \le n$, $r_2 - l_2 + 1 \ge \lfloor \frac{n}{2} \rfloor$
- Pairs (l_1, r_1) and (l_2, r_2) are distinct. That is, at least one of $l_1 \ne l_2$ and $r_1 \ne r_2$ must hold.

- Let t be the substring $s[l_1 : r_1]$ of s, and w be the substring $s[l_2 : r_2]$ of s. Then **there exists non-negative integer k, such that $f(t) = f(w) \cdot k$.**

Here substring $s[l : r]$ denotes $s_l s_{l+1} \ldots s_{r-1} s_r$, and $\lfloor x \rfloor$ denotes rounding the number down to the nearest integer.

Help Saruman solve this problem! It is guaranteed that under the constraints of the problem at least one solution exists.

Input
Each test contains multiple test cases.

The first line contains one positive integer t ($1 \le t \le 10^3$), denoting the number of test cases. Description of the test cases follows.

The first line of each test case contains one positive integer n ($2 \le n \le 2 \cdot 10^4$) — length of the string.

The second line of each test case contains a non-empty binary string of length n.

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

Output
For every test case print four integers l_1, r_1, l_2, r_2, which denote the beginning of the first substring, the end of the first substring, the beginning of the second substring, and the end of the second substring, respectively.

If there are multiple solutions, print any.

Example

input
7 6 101111 9 111000111 8 10000000 5 11011 6 001111 3 101 30 100000000000000100000000000000
output
3 6 1 3 1 9 4 9 5 8 1 4 1 5 3 5 1 6 2 4 1 2 2 3 1 15 16 30

Note
In the first testcase $f(t) = f(1111) = 15$, $f(w) = f(101) = 5$.

In the second testcase $f(t) = f(111000111) = 455$, $f(w) = f(000111) = 7$.

In the third testcase $f(t) = f(0000) = 0$, $f(w) = f(1000) = 8$.

In the fourth testcase $f(t) = f(11011) = 27$, $f(w) = f(011) = 3$.

In the fifth testcase $f(t) = f(001111) = 15$, $f(w) = f(011) = 3$.

D1. Two Hundred Twenty One (easy version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The difference between the versions is that the easy version does not require you to output the numbers of the rods to be removed. You can make hacks only if all versions of the problem are solved.

Stitch likes experimenting with different machines with his friend Sparky. Today they built another machine.

The main element of this machine are n rods arranged along one straight line and numbered from 1 to n inclusive. Each of these rods must carry an electric charge quantitatively equal to either 1 or -1 (otherwise the machine will not work). Another condition for this machine to work is that the sign-variable sum of the charge on all rods must be zero.

More formally, the rods can be represented as an array of n numbers characterizing the charge: either 1 or -1. Then the condition

must hold: $a_1 - a_2 + a_3 - a_4 + \ldots = 0$, or $\sum_{i=1}^{n} (-1)^{i-1} \cdot a_i = 0$.

Sparky charged all n rods with an electric current, but unfortunately it happened that the rods were not charged correctly (the sign-variable sum of the charge is not zero). The friends decided to leave only some of the rods in the machine. Sparky has q questions. In the ith question Sparky asks: if the machine consisted only of rods with numbers l_i to r_i inclusive, what minimal number of rods could be removed from the machine so that the sign-variable sum of charges on the remaining ones would be zero? Perhaps the friends got something wrong, and the sign-variable sum is already zero. In that case, you don't have to remove the rods at all.

If the number of rods is zero, we will assume that the sign-variable sum of charges is zero, that is, we can always remove all rods.

Help your friends and answer all of Sparky's questions!

Input

Each test contains multiple test cases.

The first line contains one positive integer t ($1 \le t \le 10^3$), denoting the number of test cases. Description of the test cases follows.

The first line of each test case contains two positive integers n and q ($1 \le n, q \le 3 \cdot 10^5$) — the number of rods and the number of questions.

The second line of each test case contains a non-empty string s of length n, where the charge of the i-th rod is 1 if s_i is the "+" symbol, or -1 if s_i is the "-" symbol.

Each next line from the next q lines contains two positive integers l_i ans r_i ($1 \le l_i \le r_i \le n$) — numbers, describing Sparky's questions.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$, and the sum of q over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print a single integer — the minimal number of rods that can be removed.

Example

input
3 14 1 +--++---++-++- 1 14 14 3 +--++---+++--- 1 14 6 12 3 10 4 10 +-+- 1 1 1 2 1 3 1 4 2 2 2 3 2 4 3 3 3 4 4 4
output
2 2 1 0 1 2 1 2 1 2 1 1 2 1

Note

In the first test case for the first query you can remove the rods numbered 5 and 8, then the following set of rods will remain: +--+---++-++-. It is easy to see that here the sign-variable sum is zero.

In the second test case:

- For the first query, we can remove the rods numbered 1 and 11, then the following set of rods will remain: --++---++---. It is easy to see that here the sign-variable sum is zero.
- For the second query we can remove the rod numbered 9, then the following set of rods will remain: ---++-. It is easy to see that here the variable sum is zero.

- For the third query we can not remove the rods at all.

D2. Two Hundred Twenty One (hard version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The difference between the versions is that the hard version does require you to output the numbers of the rods to be removed. You can make hacks only if all versions of the problem are solved.

Stitch likes experimenting with different machines with his friend Sparky. Today they built another machine.

The main element of this machine are n rods arranged along one straight line and numbered from 1 to n inclusive. Each of these rods must carry an electric charge quantitatively equal to either 1 or -1 (otherwise the machine will not work). Another condition for this machine to work is that the sign-variable sum of the charge on all rods must be zero.

More formally, the rods can be represented as an array of n numbers characterizing the charge: either 1 or -1. Then the condition must hold: $a_1 - a_2 + a_3 - a_4 + \ldots = 0$, or $\sum\limits_{i=1}^{n}(-1)^{i-1} \cdot a_i = 0$.

Sparky charged all n rods with an electric current, but unfortunately it happened that the rods were not charged correctly (the sign-variable sum of the charge is not zero). The friends decided to leave only some of the rods in the machine. Sparky has q questions. In the ith question Sparky asks: if the machine consisted only of rods with numbers l_i to r_i inclusive, what minimal number of rods could be removed from the machine so that the sign-variable sum of charges on the remaining ones would be zero? **Also Sparky wants to know the numbers of these rods**. Perhaps the friends got something wrong, and the sign-variable sum is already zero. In that case, you don't have to remove the rods at all.

If the number of rods is zero, we will assume that the sign-variable sum of charges is zero, that is, we can always remove all rods.

Help your friends and answer all of Sparky's questions!

Input

Each test contains multiple test cases.

The first line contains one positive integer t ($1 \le t \le 10^3$), denoting the number of test cases. Description of the test cases follows.

The first line of each test case contains two positive integers n and q ($1 \le n, q \le 3 \cdot 10^5$) — the number of rods and the number of questions.

The second line of each test case contains a non-empty string s of length n, where the charge of the i-th rod is 1 if s_i is the "+" symbol, or -1 if s_i is the "-" symbol.

Each next line from the next q lines contains two positive integers l_i ans r_i ($1 \le l_i \le r_i \le n$) — numbers, describing Sparky's questions.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$, and the sum of q over all test cases does not exceed $3 \cdot 10^5$.

It is guaranteed that the sum of the answers (minimal number of rods that can be removed) over all test cases does not exceed 10^6.

Output

For each test case, print the answer in the following format:

In the first line print a single integer k — the minimal number of rods that can be removed.

In the second line print k numbers separated by a space — the numbers of rods to be removed.

If there is more than one correct answer, you can print any.

Example

input
3 14 1 +--++---++-++- 1 14 14 3 +--++---+++--- 1 14 6 12 3 10 4 10 +-+- 1 1 1 2 1 3 1 4 2 2

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

output

```
2
5 8
2
1 11
1
9
0
1
1
2
1 2
1
2
2
1 3
1
2
2
2 3
1
3
1
3
2
3 4
1
4
```

Note

In the first test case for the first query you can remove the rods numbered 5 and 8, then the following set of rods will remain: `+--+---++-++-`. It is easy to see that here the sign-variable sum is zero.

In the second test case:

- For the first query, we can remove the rods numbered 1 and 11, then the following set of rods will remain: `--++---++---`. It is easy to see that here the sign-variable sum is zero.
- For the second query we can remove the rod numbered 9, then the following set of rods will remain: `---++-`. It is easy to see that here the variable sum is zero.
- For the third query we can not remove the rods at all.

E. Rescue Niwen!

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Morning desert sun horizon
Rise above the sands of time...

Fates Warning, "Exodus"

After crossing the Windswept Wastes, Ori has finally reached the Windtorn Ruins to find the Heart of the Forest! However, the ancient repository containing this priceless Willow light did not want to open!

Ori was taken aback, but the Voice of the Forest explained to him that the cunning Gorleks had decided to add protection to the repository.

The Gorleks were very fond of the "string expansion" operation. They were also very fond of increasing subsequences.

Suppose a string $s_1 s_2 s_3 \ldots s_n$ is given. Then its "expansion" is defined as the sequence of strings s_1, $s_1 s_2$, ..., $s_1 s_2 \ldots s_n$, s_2, $s_2 s_3$, ..., $s_2 s_3 \ldots s_n$, s_3, $s_3 s_4$, ..., $s_{n-1} s_n$, s_n. For example, the "expansion" the string 'abcd' will be the following sequence of strings: 'a', 'ab', 'abc', 'abcd', 'b', 'bc', 'bcd', 'c', 'cd', 'd'.

To open the ancient repository, Ori must find the size of the largest increasing subsequence of the "expansion" of the string s. Here, strings are compared lexicographically.

Help Ori with this task!

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

Each test contains multiple test cases.

The first line contains one positive integer t ($1 \le t \le 10^3$), denoting the number of test cases. Description of the test cases follows.

The first line of each test case contains one positive integer n ($1 \le n \le 5000$) — length of the string.

The second line of each test case contains a non-empty string of length n, which consists of lowercase latin letters.

It is guaranteed that the sum of n over all test cases does not exceed 10^4.

Output
For every test case print one non-negative integer — the answer to the problem.

Example

input
7 5 acbac 8 acabacba 12 aaaaaaaaaaaa 10 abacabadac 8 dcbaabcd 3 cba 6 sparky
output
9 17 12 29 14 3 9

Note
In first test case the "expansion" of the string is: 'a', 'ac', 'acb', 'acba', 'acbac', 'c', 'cb', 'cba', 'cbac', 'b', 'ba', 'bac', 'a', 'ac', 'c'. The answer can be, for example, 'a', 'ac', 'acb', 'acba', 'acbac', 'b', 'ba', 'bac', 'c'.

F. Tubular Bells

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Do you know what tubular bells are? They are a musical instrument made up of cylindrical metal tubes. In an orchestra, tubular bells are used to mimic the ringing of bells.

Mike has tubular bells, too! They consist of n tubes, and each of the tubes has a length that can be expressed by a integer from l to r inclusive. It is clear that the lengths of all the tubes are different (it makes no sense to make the same tubes). It is also known that $r - l + 1 = n$.

Formally, we can say that Mike's tubular bells are described by a permutation a of length n that contains all numbers from l to r inclusive, with a_i denoting the length of the i-th tube.

You are offered an interesting task: to guess what Mike's instrument looks like. Simply, you must guess the permutation.

Mike won't tell you l or r. He will only tell you n, and will allow you to ask no more than $n + 5000$ queries.

In each query, you name two positive integers x, y such that $1 \le x, y \le n, x \ne y$. In response to this query, the program written by Mike will give you $\mathrm{lcm}(a_x, a_y)$, where $\mathrm{lcm}(c, d)$ denotes the least common multiple of c and d.

Solve Mike's problem!

Input
Each test contains multiple test cases.

The first line contains one positive integer t ($1 \le t \le 20$), denoting the number of test cases. Description of the test cases follows.

The single line of each test case contains one positive integer n ($3 \le n \le 10^5$) — number of tubes in Mike's tubular bells. Also $1 \le l \le r \le 2 \cdot 10^5$, i.e. the lengths of the tubes do not exceed $2 \cdot 10^5$.

It is guaranteed that the sum of **maximal number of queries** (i.e. $n + 5000$) over all test cases does not exceed $10^5 + 5000$. It means that sum of n does not exceed $10^5 + 5000 - t \cdot 5000$.

Interaction
For each set of input data, read one integer n. You are allowed to make no more than $n + 5000$ queries.

If you want to make a query, output it in the format "? x y", where x and y — the numbers of tubes for which you learn the lcm (least common multiple) of their lengths. Note that $1 \le x, y \le n, x \ne y$ must be satisfied. The interactor will return a single integer —the answer to your query.

If you are ready to print the answer, print it in the format "! a_1 a_2 ... a_n". **The output of the answer is not considered a query and is not included in the number of queries.**

After each query and answer output, don't forget to output the line translation and reset the output buffer. Otherwise you will get the verdict "Idleness limit exceeded". To reset the buffer use:

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

Note that the interactor is not adaptive. That is, the original permutation is fixed in the beginning and don't depend on your queries.

Hacks:

Use the following format for hacks:

The first line contains a single positive integer t ($1 \le t \le 20$) — the number of input datasets. A description of the input data sets is given below.

The first line of each test case contains one positive integer n ($3 \le n \le 10^5$) —the number of tubes. It is known that $1 \le l \le r \le 2 \cdot 10^5$, i.e. the lengths of the tubes do not exceed $2 \cdot 10^5$.

The second line of each test case contains an array a of n positive integers — the lengths of the tubes in each input dataset. Remember that $l \le a_i \le r$ and $r - l + 1 = n$, and that all a_i are different.

Example

input
3 5 8 10 7 6 9 5 24 25 28 27 26 7 1 2 3 4 5 6 7
output
? 1 2 40 ? 2 5 90 ? 3 1 56 ? 4 5 18 ! 8 10 7 6 9 ? 1 5 312 ? 2 4 675 ! 24 25 28 27 26 ? 1 4 4 ? 2 5 10 ? 3 7 21 ? 6 2 6 ? 2 5 10 ? 1 2 2 ? 1 2 2 ? 1 2 2 ? 1 2 2 ? 1 2 2 ! 1 2 3 4 5 6 7