Codeforces Round #734 (Div. 3)

A. Polycarp and Coins

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

Polycarp must pay **exactly** n burles at the checkout. He has coins of two nominal values: 1 burle and 2 burles. Polycarp likes both kinds of coins equally. So he doesn't want to pay with more coins of one type than with the other.

Thus, Polycarp wants to minimize the difference between the count of coins of 1 burle and 2 burles being used. Help him by determining two non-negative integer values c_1 and c_2 which are the number of coins of 1 burle and 2 burles, respectively, so that the total value of that number of coins is **exactly** n (i. e. $c_1 + 2 \cdot c_2 = n$), and the absolute value of the difference between c_1 and c_2 is as little as possible (i. e. you must minimize $|c_1 - c_2|$).

Input

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

Each test case consists of one line. This line contains one integer n ($1 \le n \le 10^9$) — the number of burles to be paid by Polycarp.

Output

For each test case, output a separate line containing two integers c_1 and c_2 ($c_1, c_2 \ge 0$) separated by a space where c_1 is the number of coins of 1 burle and c_2 is the number of coins of 2 burles. If there are multiple optimal solutions, print any one.

Example

input

```
6
1000
30
1
32
1000000000
5
```

output

```
334 333
10 10
1 0
10 11
333333334 333333333
1 2
```

Note

The answer for the first test case is "334 333". The sum of the nominal values of all coins is $334 \cdot 1 + 333 \cdot 2 = 1000$, whereas $|334 - 333| = 1$. One can't get the better value because if $|c_1 - c_2| = 0$, then $c_1 = c_2$ and $c_1 \cdot 1 + c_1 \cdot 2 = 1000$, but then the value of c_1 isn't an integer.

The answer for the second test case is "`10 10`". The sum of the nominal values is $10 \cdot 1 + 10 \cdot 2 = 30$ and $|10 - 10| = 0$, whereas there's no number having an absolute value less than 0.

B1. Wonderful Coloring - 1

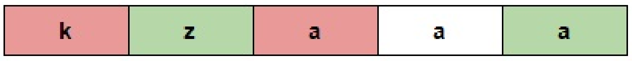
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is a simplified version of the problem B2. Perhaps you should read the problem B2 before you start solving B1.

Paul and Mary have a favorite string s which consists of lowercase letters of the Latin alphabet. They want to paint it using pieces of chalk of two colors: red and green. Let's call a coloring of a string wonderful if the following conditions are met:

1. each letter of the string is either painted in exactly one color (red or green) or isn't painted;
2. each two letters which are painted in the same color are different;
3. the number of letters painted in red is equal to the number of letters painted in green;
4. the number of painted letters of this coloring is **maximum** among all colorings of the string which meet the first three conditions.

E. g. consider a string s equal to "kzaaa". One of the wonderful colorings of the string is shown in the figure.



The example of a wonderful coloring of the string "kzaaa".

Paul and Mary want to learn by themselves how to find a wonderful coloring of the string. But they are very young, so they need a hint. Help them find k — the number of red (or green, these numbers are equal) letters in a wonderful coloring.

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

Each test case consists of one non-empty string s which consists of lowercase letters of the Latin alphabet. The number of characters in the string doesn't exceed 50.

Output

For each test case, output a separate line containing one non-negative integer k — the number of letters which will be painted in red in a wonderful coloring.

Example

input

```
5
kzaaa
codeforces
archive
y
xxxxxx
```

output

```
2
5
3
0
1
```

Note

The first test case contains the string from the statement. One of the wonderful colorings is shown in the figure. There's no wonderful coloring containing 3 or more red letters because the total number of painted symbols will exceed the string's length.

The string from the second test case can be painted as follows. Let's paint the first occurrence of each of the letters "c", "o", "e" in red and the second ones in green. Let's paint the letters "d", "f" in red and "r", "s" in green. So every letter will be painted in red or green, hence the answer better than 5 doesn't exist.

The third test case contains the string of distinct letters, so you can paint any set of characters in red, as long as the size of this set doesn't exceed half of the size of the string and is the maximum possible.

The fourth test case contains a single letter which cannot be painted in red because there will be no letter able to be painted in green.

The fifth test case contains a string of identical letters, so there's no way to paint more than one letter in red.

B2. Wonderful Coloring - 2

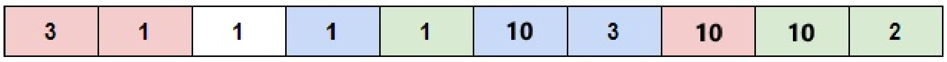
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is an extension of the problem "Wonderful Coloring - 1". It has quite many differences, so you should read this statement completely.

Recently, Paul and Mary have found a new favorite sequence of integers $a_1, a_2, \dots, a_n$. They want to paint it using pieces of chalk of k colors. The coloring of a sequence is called *wonderful* if the following conditions are met:

1. each element of the sequence is either painted in one of k colors or isn't painted;
2. each two elements which are painted in the same color are different (i. e. there's no two equal values painted in the same color);
3. let's calculate for each of k colors the number of elements painted in the color — all calculated numbers must be equal;
4. the total number of painted elements of the sequence is the **maximum** among all colorings of the sequence which meet the first three conditions.

E. g. consider a sequence $a = [3, 1, 1, 1, 1, 10, 3, 10, 10, 2]$ and $k = 3$. One of the wonderful colorings of the sequence is shown in the figure.



The example of a wonderful coloring of the sequence $a = [3, 1, 1, 1, 1, 10, 3, 10, 10, 2]$ and $k = 3$. Note that one of the elements isn't painted.

Help Paul and Mary to find a wonderful coloring of a given sequence a.

Input

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

Each test case consists of two lines. The first one contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le n$) — the length of a given sequence and the number of colors, respectively. The second one contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$).

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

Output

Output t lines, each of them must contain a description of a wonderful coloring for the corresponding test case.

Each wonderful coloring must be printed as a sequence of n integers $c_1, c_2, \dots, c_n$ ($0 \le c_i \le k$) separated by spaces where

- $c_i = 0$, if i-th element isn't painted;
- $c_i > 0$, if i-th element is painted in the c_i-th color.

Remember that you need to maximize the total count of painted elements for the wonderful coloring. If there are multiple solutions, print any one.

Example

input

```
6
10 3
3 1 1 1 1 10 3 10 10 2
4 4
1 1 1 1
1 1
1
13 1
3 1 4 1 5 9 2 6 5 3 5 8 9
13 2
3 1 4 1 5 9 2 6 5 3 5 8 9
13 3
3 1 4 1 5 9 2 6 5 3 5 8 9
```

output

```
1 1 0 2 3 2 2 1 3 3
4 2 1 3
1
0 0 1 1 0 1 1 1 0 1 1 1 0
2 1 2 2 1 1 1 1 2 1 0 2 2
1 1 3 2 1 3 3 1 2 2 3 2 0
```

Note

In the first test case, the answer is shown in the figure in the statement. The red color has number 1, the blue color — 2, the green — 3.

C. Interesting Story

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Stephen Queen wants to write a story. He is a very unusual writer, he uses only letters 'a', 'b', 'c', 'd' and 'e'!

To compose a story, Stephen wrote out n words consisting of the first 5 lowercase letters of the Latin alphabet. He wants to select the **maximum** number of **words** to make an **interesting** story.

Let a story be a sequence of words that are not necessarily different. A story is called *interesting* if there exists a letter which occurs among all words of the story more times than all other letters together.

For example, the story consisting of three words "bac", "aaada", "e" is interesting (the letter 'a' occurs 5 times, all other letters occur 4 times in total). But the story consisting of two words "aba", "abcde" is not (no such letter that it occurs more than all other letters in total).

You are given a sequence of n words consisting of letters 'a', 'b', 'c', 'd' and 'e'. Your task is to choose the maximum number of them to make an interesting story. If there's no way to make a non-empty story, output 0.

Input

The first line contains one integer t ($1 \le t \le 5000$) — the number of test cases. Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of the words in the sequence. Then n lines follow, each of them contains a word — a non-empty string consisting of lowercase letters of the Latin alphabet. The words in the sequence may be non-distinct (i. e. duplicates are allowed). Only the letters 'a', 'b', 'c', 'd' and 'e' may occur in the words.

It is guaranteed that the sum of n over all test cases doesn't exceed $2 \cdot 10^5$; the sum of the lengths of all words over all test cases doesn't exceed $4 \cdot 10^5$.

Output

For each test case, output the maximum number of words that compose an interesting story. Print `0` if there's no way to make a non-empty interesting story.

Example

input

```
6
3
bac
aaada
e
3
aba
abcde
aba
2
baba
baba
4
ab
ab
c
bc
5
cbdca
d
a
d
e
3
b
c
ca
```

output

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

Note

In the first test case of the example, all 3 words can be used to make an interesting story. The interesting story is "bac aaada e".

In the second test case of the example, the 1-st and the 3-rd words can be used to make an interesting story. The interesting story is "aba aba". Stephen can't use all three words at the same time.

In the third test case of the example, Stephen can't make a non-empty interesting story. So the answer is 0.

In the fourth test case of the example, Stephen can use the 3-rd and the 4-th words to make an interesting story. The interesting story is "c bc".

D1. Domino (easy version)

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

The only difference between this problem and D2 is that you don't have to provide the way to construct the answer in this problem, but you have to do it in D2.

There's a table of $n \times m$ cells (n rows and m columns). The value of $n \cdot m$ is even.

A domino is a figure that consists of two cells having a common side. It may be horizontal (one of the cells is to the right of the other) or vertical (one of the cells is above the other).

You need to find out whether it is possible to place $\frac{nm}{2}$ dominoes on the table so that exactly k of them are horizontal and all the other dominoes are vertical. The dominoes cannot overlap and must fill the whole table.

Input

The first line contains one integer t $(1 \le t \le 10)$ — the number of test cases. Then t test cases follow.

Each test case consists of a single line. The line contains three integers n, m, k ($1 \le n, m \le 100$, $0 \le k \le \frac{nm}{2}$, $n \cdot m$ is even) — the number of rows, columns and horizontal dominoes, respectively.

Output

For each test case output "YES", if it is possible to place dominoes in the desired way, or "NO" otherwise.

You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example

input

```
8
4 4 2
2 3 0
3 2 3
1 2 0
2 4 2
5 2 2
2 17 16
2 1 1
```

output

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

D2. Domino (hard version)

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

The only difference between this problem and D1 is that you don't have to provide the way to construct the answer in D1, but you have to do it in this problem.

There's a table of $n \times m$ cells (n rows and m columns). The value of $n \cdot m$ is even.

A domino is a figure that consists of two cells having a common side. It may be horizontal (one of the cells is to the right of the other) or vertical (one of the cells is above the other).

You need to place $\frac{nm}{2}$ dominoes on the table so that exactly k of them are horizontal and all the other dominoes are vertical. The dominoes cannot overlap and must fill the whole table.

Input

The first line contains one integer t ($1 \le t \le 10$) — the number of test cases. Then t test cases follow.

Each test case consists of a single line. The line contains three integers n, m, k ($1 \le n, m \le 100$, $0 \le k \le \frac{nm}{2}$, $n \cdot m$ is even) — the count of rows, columns and horizontal dominoes, respectively.

Output

For each test case:

- print "NO" if it's not possible to place the dominoes on the table in the described way;
- otherwise, print "YES" on a separate line, then print n lines so that each of them contains m lowercase letters of the Latin alphabet — the layout of the dominoes on the table. Each cell of the table must be marked by the letter so that for every two cells having a common side, they are marked by the same letters if and only if they are occupied by the same domino. I.e. both cells of the same domino must be marked with the same letter, but two dominoes that share a side must be marked with different letters. If there are multiple solutions, print any of them.

Example

input

```
8
4 4 2
2 3 0
3 2 3
1 2 0
2 4 2
5 2 2
2 17 16
2 1 1
```

output

```
YES
accx
aegx
bega
bdda
YES
aha
aha
YES
zz
aa
zz
NO
YES
```

```
aaza
bbza
NO
YES
bbaabbaabbaabbaay
ddccddccddccddccy
NO
```

E. Fixed Points

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Consider a sequence of integers $a_1, a_2, \ldots, a_n$. In one move, you can select any element of the sequence and delete it. After an element is deleted, all elements to the right are shifted to the left by 1 position, so there are no empty spaces in the sequence. So after you make a move, the sequence's length decreases by 1. The indices of the elements after the move are recalculated.

E. g. let the sequence be $a = [3, 2, 2, 1, 5]$. Let's select the element $a_3 = 2$ in a move. Then after the move the sequence will be equal to $a = [3, 2, 1, 5]$, so the 3-rd element of the new sequence will be $a_3 = 1$ and the 4-th element will be $a_4 = 5$.

You are given a sequence $a_1, a_2, \ldots, a_n$ and a number k. You need to find the minimum number of moves you have to make so that in the resulting sequence there will be **at least** k elements that are equal to their indices, i. e. the resulting sequence $b_1, b_2, \ldots, b_m$ will contain at least k indices i such that $b_i = i$.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

Each test case consists of two consecutive lines. The first line contains two integers n and k ($1 \le k \le n \le 2000$). The second line contains a sequence of integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$). The numbers in the sequence are not necessarily different.

It is guaranteed that the sum of n over all test cases doesn't exceed 2000.

Output

For each test case output in a single line:

- -1 if there's no desired move sequence;
- otherwise, the integer x ($0 \le x \le n$) — the minimum number of the moves to be made so that the resulting sequence will contain at least k elements that are equal to their indices.

Example

input

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

output

```
1
2
-1
2
```

Note

In the first test case the sequence doesn't satisfy the desired condition, but it can be provided by deleting the first element, hence the sequence will be $[1, 2, 3, 4, 5, 6]$ and 6 elements will be equal to their indices.

In the second test case there are two ways to get the desired result in 2 moves: the first one is to delete the 1-st and the 3-rd elements so that the sequence will be $[1, 2, 3]$ and have 3 elements equal to their indices; the second way is to delete the 2-nd and the 3-rd elements to get the sequence $[5, 2, 3]$ with 2 desired elements.

F. Equidistant Vertices

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

A tree is an undirected connected graph without cycles.

You are given a tree of n vertices. Find the number of ways to choose exactly k vertices in this tree (i. e. a k-element subset of vertices) so that all pairwise distances between the selected vertices are equal (in other words, there exists an integer c such that for all u, v ($u \neq v$, u, v are in selected vertices) $d_{u,v} = c$, where $d_{u,v}$ is the distance from u to v).

Since the answer may be very large, you need to output it modulo $10^9 + 7$.

Input

The first line contains one integer t ($1 \le t \le 10$) — the number of test cases. Then t test cases follow.

Each test case is preceded by an empty line.

Each test case consists of several lines. The first line of the test case contains two integers n and k ($2 \le k \le n \le 100$) — the number of vertices in the tree and the number of vertices to be selected, respectively. Then $n - 1$ lines follow, each of them contains two integers u and v ($1 \le u, v \le n$, $u \neq v$) which describe a pair of vertices connected by an edge. It is guaranteed that the given graph is a tree and has no loops or multiple edges.

Output

For each test case output in a separate line a single integer — the number of ways to select exactly k vertices so that for all pairs of selected vertices the distances between the vertices in the pairs are equal, modulo $10^9 + 7$ (in other words, print the remainder when divided by 1000000007).

Example

input

```
3

4 2
1 2
2 3
2 4

3 3
1 2
2 3

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

output

```
6
0
1
```