Round E APAC Test 2017

A. Diwali lightings

B. Beautiful Numbers

C. Partioning Number

D. Sorting Array

Questions asked 3

Submissions

Diwali lightings

5pt Not attempted **1615/2160 users** correct (75%)

8pt Not attempted **1262/1580 users** correct (80%)

Beautiful Numbers

6pt Not attempted **1429/1592 users** correct (90%)

15pt Not attempted **211/1189 users** correct (18%)

Partioning Number

9pt Not attempted **646/851 users** correct (76%)

17pt Not attempted **193/470 users** correct (41%)

Sorting Array

13pt Not attempted **5/65 users** correct (8%)

27pt Not attempted **2/2 users** correct (100%)

Top Scores

AngryBacon	100
LittleBuger	100
wcwswswws	78
legedexinshi	73
TheTerminalGuy	71
Shaon	71
ajs97	65
thonsi	65
john0312	65
rossSJTU	65

Problem A. Diwali lightings

Small input 5 points	Solve A-small
Large input 8 points	Solve A-large

Problem

Diwali is the festival of lights. To celebrate it, people decorate their houses with multi-color lights and burst crackers. Everyone loves Diwali, and so does Pari. Pari is very fond of lights, and has transfinite powers, so she buys an infinite number of red and blue light bulbs. As a programmer, she also loves patterns, so she arranges her lights by infinitely repeating a given finite pattern **S**.

For example, if **S** is BBRB, the infinite sequence Pari builds would be BBRBBBRBBBRB...

Blue is Pari's favorite color, so she wants to know the number of blue bulbs between the **I**th bulb and **J**th bulb, inclusive, in the infinite sequence she built (lights are numbered with consecutive integers starting from 1). In the sequence above, the indices would be numbered as follows:

```
B  B  R  B  B  B  R  B  B  B  R  B...
1  2  3  4  5  6  7  8  9  10 11 12
```

So, for example, there are 4 blue lights between the 4th and 8th positions, but only 2 between the 10th and 12th.

Since the sequence can be very long, she wrote a program to do the count for her. Can you do the same?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.
First line of each test case consists of a string S, denoting the initial finite pattern.
Second line of each test case consists of two space separated integers **I** and **J**, defined above.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is number of blue bulbs between the **I**th bulb and **J**th bulb of Pari's infinite sequence, inclusive.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le$ length of $\mathbf{S} \le 100$.
Each character of **S** is either uppercase B or uppercase R.

Small dataset

$1 \le \mathbf{I} \le \mathbf{J} \le 10^6$.

Large dataset

$1 \le \mathbf{I} \le \mathbf{J} \le 10^{18}$.

Sample

Input	Output
3 BBRB 4 8 BBRB 10 12 BR 1 1000000	Case #1: 4 Case #2: 2 Case #3: 500000

Cases #1 and #2 are explained above.

In Case #3, bulbs at odd indices are always blue, and bulbs at even indices are always red, so there are half a million blue bulbs between positions 1 and 10^6.

Submissions

Diwali lightings

5pt	Not attempted **1615/2160 users** correct (75%)
8pt	Not attempted **1262/1580 users** correct (80%)

Beautiful Numbers

6pt	Not attempted **1429/1592 users** correct (90%)
15pt	Not attempted **211/1189 users** correct (18%)

Partioning Number

9pt	Not attempted **646/851 users** correct (76%)
17pt	Not attempted **193/470 users** correct (41%)

Sorting Array

13pt	Not attempted **5/65 users** correct (8%)
27pt	Not attempted **2/2 users** correct (100%)

Top Scores

AngryBacon	100
LittleBuger	100
wcwswswws	78
legedexinshi	73
TheTerminalGuy	71
Shaon	71
ajs97	65
thonsi	65
john0312	65
rossSJTU	65

Problem B. **Beautiful Numbers**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 6 points	Solve B-small
Large input 15 points	Solve B-large

Problem

We consider a number to be *beautiful* if it consists only of the digit 1 repeated one or more times. Not all numbers are beautiful, but we can make any base 10 positive integer beautiful by writing it in another base.

Given an integer **N**, can you find a base *B* (with *B* > 1) to write it in such that all of its digits become 1? If there are multiple bases that satisfy this property, choose the one that maximizes the number of 1 digits.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with an integer **N**.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the base described in the problem statement.

Limits

1 ≤ **T** ≤ 100.

Small dataset

3 ≤ **N** ≤ 1000.

Large dataset

3 ≤ **N** ≤ 10^{18}.

Sample

Input	Output
2 3 13	Case #1: 2 Case #2: 3

In case #1, the optimal solution is to write 3 as 11 in base 2.

In case #2, the optimal solution is to write 13 as 111 in base 3. Note that we could also write 13 as 11 in base 12, but neither of those representations has as many 1s.

Submissions

Diwali lightings

5pt Not attempted **1615/2160 users** correct (75%)

8pt Not attempted **1262/1580 users** correct (80%)

Beautiful Numbers

6pt Not attempted **1429/1592 users** correct (90%)

15pt Not attempted **211/1189 users** correct (18%)

Partioning Number

9pt Not attempted **646/851 users** correct (76%)

17pt Not attempted **193/470 users** correct (41%)

Sorting Array

13pt Not attempted **5/65 users** correct (8%)

27pt Not attempted **2/2 users** correct (100%)

Top Scores

AngryBacon	100
LittleBuger	100
wcwswswws	78
legedexinshi	73
TheTerminalGuy	71
Shaon	71
ajs97	65
thonsi	65
john0312	65
rossSJTU	65

Problem C. **Partioning Number**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 9 points	Solve C-small
Large input 17 points	Solve C-large

Problem

Shekhu has **N** balls. She wants to distribute them among one or more buckets in a way that satisfies all of these constraints:

1. The numbers of balls in the buckets must be in non-decreasing order when read from left to right.
2. The leftmost bucket must be non-empty and the number of balls in the leftmost bucket must be divisible by **D**.
3. The difference (in number of balls) between *any* two buckets (not just any two adjacent buckets) must be less than or equal to 2.

How many different ways are there for Shekhu to do this? Two ways are considered different if the lists of numbers of balls in buckets, reading left to right, are different.

Input

The first line of the input gives the number of test cases, **T**.
T test cases follow. Each test case consists of one line with two integers **N** and **D**, as described above.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the answer, as described above.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{D} \le 100$.

Small dataset

$1 \le \mathbf{N} \le 2000$.

Large dataset

$1 \le \mathbf{N} \le 10^5$.

Sample

Input	Output
3 7 1 7 2 2 4	Case #1: 10 Case #2: 1 Case #3: 0

In sample case #1, the possible distributions are:

- 1 1 1 1 1 1 1
- 1 1 1 1 1 2
- 1 1 1 1 3
- 1 1 1 2 2
- 1 2 2 2
- 1 1 2 3
- 1 3 3
- 2 2 3
- 3 4
- 7

Note that 1 2 4 is not a valid distribution, since the difference between 1 and 4 is more than 2.

In sample case #2, the possible distributions are:

- 2 2 3

3 4 is not possible, since the first term is not divisible by 2.

In sample case #3, no possible arrangement exists.

Problem D. Sorting Array

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 13 points	Solve D-small
Large input 27 points	Solve D-large

Problem

We are in the process of creating a somehow esoteric sorting algorithm to sort an array A of all integers between 1 and **N**. The integers in A can start in an arbitrary order. Besides the input order, the algorithm depends on two integers **P**(which would be at most 3) and **K**. Here is how the algorithms works:

1. Partition A into **K** disjoint non-empty subarrays A_1, A_2, ..., A_K such that concatenating them in order A_1A_2 ... A_K produces A.
2. Sort each subarray individually.
3. Choose up to **P** of the subarrays, and swap any two of them any number of times.

For example, consider A = [1 5 4 3 2] and **P** = 2. A possible partition into **K** = 4 disjoint subarrays is:

```
A1 = [1]
A2 = [5]
A3 = [4]
A4 = [3 2]

After Sorting Each Subarray:

A1 = [1]
A2 = [5]
A3 = [4]
A4 = [2 3]

After swapping A4 and A2:

A1 = [1]
A2 = [2 3]
A3 = [4]
A4 = [5]
```

We want to show the algorithm is good for distributed environments by finding, for a fixed input and value of **P**, the maximum number of partitions **K** such that, choosing the partitions and swaps wisely, we can achieve a sorting of the original order. Can you help us to calculate that **K**?

Input

The first line of the input gives the number of test cases, **T**.
T test cases follow. Each test case consists of two lines. The first line contains two integers **N** and **P**, as described above.
The second line of the test case contains **N** integers $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_N}$ represting array A.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the maximum possible value for the parameter **K**.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 5000$.
$1 \le \mathbf{X_i} \le \mathbf{N}$, for all i.
$\mathbf{X_i} \ne \mathbf{X_j}$ for all $i \ne j$.

Small dataset

P = 2.

Large dataset

P = 3.

Sample

Input	Output

<pre>5
5 2
1 5 4 3 2
5 2
4 5 1 2 3
6 2
6 3 5 2 4 1
5 3
4 5 1 2 3
6 3
1 2 6 4 5 3</pre>
	<pre>Case #1: 4
	Case #2: 2
	Case #3: 3
	Case #4: 3
	Case #5: 6</pre>

Case #1:
Same as walk through in the statement.

Case #2:
[4 5] [1 2 3]
Swap the 2 blocks: [1 2 3] [4 5]

Case #3:
[6] [3 5 2 4] [1]
Sort [3 5 2 4], then swap [6] and [1], we get: [1] [2 3 4 5] [6]

Case #4:
[4 5] [1] [2 3]
Swap [4 5] and [1], then swap [2 3] and [4 5]: [1] [2 3] [4 5]

Case #5:
[1] [2] [6] [4] [5] [3]
Swap [6] and [3]: [1] [2] [3] [4] [5] [6]

Note: First 3 sample cases would not appear in the Large dataset and the last 2 sample cases would not appear in the Small dataset.