Code Jam 2021 - Qualification Round

Reversort

Problem

Note: The main parts of the statements of the problems "Reversort" and "Reversort Engineering" are identical, except for the last paragraph. The problems can otherwise be solved independently.

Reversort is an algorithm to sort a list of distinct integers in increasing order. The algorithm is based on the "Reverse" operation. Each application of this operation reverses the order of some contiguous part of the list.

The pseudocode of the algorithm is the following:

```
Reversort(L):
  for i := 1 to length(L) - 1
    j := position with the minimum value in L between i and length(L), inclusive
    Reverse(L[i..j])
```

After $i-1$ iterations, the positions $1,\ 2,\ \ldots,\ i-1$ of the list contain the $i-1$ smallest elements of L, in increasing order. During the i-th iteration, the process reverses the sublist going from the i-th position to the current position of the i-th minimum element. That makes the i-th minimum element end up in the i-th position.

For example, for a list with 4 elements, the algorithm would perform 3 iterations. Here is how it would process $L = [4, 2, 1, 3]$:

1. $i = 1,\ j = 3 \longrightarrow L = [1, 2, 4, 3]$
2. $i = 2,\ j = 2 \longrightarrow L = [1, 2, 4, 3]$
3. $i = 3,\ j = 4 \longrightarrow L = [1, 2, 3, 4]$

The most expensive part of executing the algorithm on our architecture is the Reverse operation. Therefore, our measure for the cost of each iteration is simply the length of the sublist passed to Reverse, that is, the value $j - i + 1$. The cost of the whole algorithm is the sum of the costs of each iteration.

In the example above, the iterations cost 3, 1, and 2, in that order, for a total of 6.

Given the initial list, compute the cost of executing Reversort on it.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case consists of 2 lines. The first line contains a single integer $\mathbf{N}$, representing the number of elements in the input list. The second line contains $\mathbf{N}$ distinct integers $\mathbf{L_1}$, $\mathbf{L_2}$, ..., $\mathbf{L_N}$, representing the elements of the input list L, in order.

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 total cost of executing Reversort on the list given as input.

Limits

Time limit: 10 seconds.
Memory limit: 1 GB.

Test Set 1 (Visible Verdict)

$1 \leq \mathbf{T} \leq 100$.
$2 \leq \mathbf{N} \leq 100$.
$1 \leq \mathbf{L_i} \leq N$, for all i.
$\mathbf{L_i} \neq \mathbf{L_j}$, for all $i \neq j$.

Sample

Sample Input

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

Sample Output

```
Case #1: 6
Case #2: 1
Case #3: 12
```

Sample Case #1 is described in the statement above.

In Sample Case #2, there is a single iteration, in which Reverse is applied to a sublist of size 1. Therefore, the total cost is 1.

In Sample Case #3, the first iteration reverses the full list, for a cost of 7. After that, the list is already sorted, but there are 5 more iterations, each of which contributes a cost of 1.

Code Jam 2021 - Qualification Round

Moons and Umbrellas

Problem

Cody-Jamal is working on his latest piece of abstract art: a mural consisting of a row of waning moons and closed umbrellas. Unfortunately, greedy copyright trolls are claiming that waning moons look like an uppercase C and closed umbrellas look like a J, and they have a copyright on CJ and JC. Therefore, for each time CJ appears in the mural, Cody-Jamal must pay $\mathbf{X}$, and for each time JC appears in the mural, he must pay $\mathbf{Y}$.



Cody-Jamal is unwilling to let them compromise his art, so he will not change anything already painted. He decided, however, that the empty spaces he still has could be filled strategically, to minimize the copyright expenses.

For example, if `CJ?CC?` represents the current state of the mural, with `C` representing a waning moon, `J` representing a closed umbrella, and `?` representing a space that still needs to be painted with either a waning moon or a closed umbrella, he could finish the mural as `CJCCCC`, `CJCCCJ`, `CJJCCC`, or `CJJCCJ`. The first and third options would require paying $\mathbf{X} + \mathbf{Y}$ in copyrights, while the second and fourth would require paying $2 \cdot \mathbf{X} + \mathbf{Y}$.

Given the costs $\mathbf{X}$ and $\mathbf{Y}$ and a string representing the current state of the mural, how much does Cody-Jamal need to pay in copyrights if he finishes his mural in a way that minimizes that cost?

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ lines follow. Each line contains two integers $\mathbf{X}$ and $\mathbf{Y}$ and a string $\mathbf{S}$ representing the two costs and the current state of the mural, respectively.

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 minimum cost that Cody-Jamal needs to pay in copyrights for a finished mural.

Limits

Time limit: 10 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
Each character of $\mathbf{S}$ is either `C`, `J`, or `?`.

Test Set 1 (Visible Verdict)

$1 \leq$ the length of $\mathbf{S} \leq 10$.
$1 \leq \mathbf{X} \leq 100$.
$1 \leq \mathbf{Y} \leq 100$.

Test Set 2 (Visible Verdict)

$1 \leq$ the length of $\mathbf{S} \leq 1000$.
$1 \leq \mathbf{X} \leq 100$.
$1 \leq \mathbf{Y} \leq 100$.

Extra credit!

What if some copyright holders could pay Cody-Jamal for the advertisement instead of being paid? Cody-Jamal getting paid is represented by a negative cost.

Test Set 3 (Hidden Verdict)

$1 \leq$ the length of $\mathbf{S} \leq 1000$.
$-100 \leq \mathbf{X} \leq 100$.
$-100 \leq \mathbf{Y} \leq 100$.

Sample

Note: there are additional samples that are not run on submissions down below.

Sample Input	Sample Output
4 2 3 CJ?CC? 4 2 CJCJ 1 3 C?J 2 5 ??J???	Case #1: 5 Case #2: 10 Case #3: 1 Case #4: 0

Sample Case #1 is the one explained in the problem statement. The minimum cost is $\mathbf{X} + \mathbf{Y} = 2 + 3 = 5$.

In Sample Case #2, Cody-Jamal is already finished, so he does not have a choice. There are two `CJ`s and one `JC` in his mural.

In Sample Case #3, substituting either `C` or `J` results in one `CJ` either from the second and third characters or the first and second characters, respectively.

In Sample Case #4, Cody-Jamal can finish his mural with all `J`s. Since that contains no instance of `CJ` nor `JC`, it yields no copyright cost.

Additional Sample - Test Set 3

The following additional sample fits the limits of Test Set 3. It will not be run against your submitted solutions.

Sample Input	Sample Output

```
1
2 -5 ??JJ??
```

```
Case #1: -8
```

In Sample Case #1 for Test Set 3, Cody-Jamal can finish his mural optimally as `JCJJCC` or `JCJJJC`. Either way, there is one `CJ` and two `JC`s in his mural.

Code Jam 2021 - Qualification Round

Reversort Engineering

Problem

Note: The main parts of the statements of the problems "Reversort" and "Reversort Engineering" are identical, except for the last paragraph. The problems can otherwise be solved independently.

Reversort is an algorithm to sort a list of distinct integers in increasing order. The algorithm is based on the "Reverse" operation. Each application of this operation reverses the order of some contiguous part of the list.

The pseudocode of the algorithm is the following:

```
Reversort(L):
  for i := 1 to length(L) - 1
    j := position with the minimum value in L between i and length(L), inclusive
    Reverse(L[i..j])
```

After $i-1$ iterations, the positions $1,\ 2,\ \ldots,\ i-1$ of the list contain the $i-1$ smallest elements of L, in increasing order. During the i-th iteration, the process reverses the sublist going from the i-th position to the current position of the i-th minimum element. That makes the i-th minimum element end up in the i-th position.

For example, for a list with 4 elements, the algorithm would perform 3 iterations. Here is how it would process $L = [4, 2, 1, 3]$:

1. $i = 1,\ j = 3 \longrightarrow L = [1, 2, 4, 3]$
2. $i = 2,\ j = 2 \longrightarrow L = [1, 2, 4, 3]$
3. $i = 3,\ j = 4 \longrightarrow L = [1, 2, 3, 4]$

The most expensive part of executing the algorithm on our architecture is the Reverse operation. Therefore, our measure for the cost of each iteration is simply the length of the sublist passed to Reverse, that is, the value $j - i + 1$. The cost of the whole algorithm is the sum of the costs of each iteration.

In the example above, the iterations cost 3, 1, and 2, in that order, for a total of 6.

You are given a size $\mathbf{N}$ and a cost $\mathbf{C}$. Find a list of $\mathbf{N}$ distinct integers between 1 and $\mathbf{N}$ such that the cost of applying Reversort to it is exactly $\mathbf{C}$, or say that there is no such list.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ lines follow. Each line describes a test case with two integers $\mathbf{N}$ and $\mathbf{C}$, the size of the wanted list and the desired cost, respectively.

Output

For each test case, if there is no list of size $\mathbf{N}$ such that applying Reversort to it costs exactly $\mathbf{C}$, output one line containing `Case #`x`: IMPOSSIBLE`, where x is the test case number (starting from 1). Otherwise, output one line containing `Case #`x`:` y_1 y_2 `...` $y_\mathbf{N}$, where x is the test case number (starting from 1) and each y_i is a distinct integer between 1 and $\mathbf{N}$, representing the i-th element of one such possible list.

If there are multiple solutions, you may output any one of them. (See "What if a test case has multiple correct solutions?" in the Competing section of the FAQ.) This information about multiple solutions will not be explicitly stated in the remainder of the 2021 contest.

Limits

Time limit: 10 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
$1 \leq \mathbf{C} \leq 1000$.

Test Set 1 (Visible Verdict)

$2 \leq \mathbf{N} \leq 7$.

Test Set 2 (Visible Verdict)

$2 \leq \mathbf{N} \leq 100$.

Sample

Sample Input

```
5
4 6
2 1
7 12
7 2
2 1000
```

Sample Output

```
Case #1: 4 2 1 3
Case #2: 1 2
Case #3: 7 6 5 4 3 2 1
Case #4: IMPOSSIBLE
Case #5: IMPOSSIBLE
```

Sample Case #1 is described in the statement above.

In Sample Case #2, the algorithm runs for only one iteration on the proposed output. In that iteration, reverse is applied to a sublist of size 1, therefore, its cost is 1.

In Sample Case #3, the first iteration reverses the full list, for a cost of 7. After that, the list is already sorted, but there are 5 more iterations, each of which contributes a cost of 1. Another valid output would be `7 5 4 3 2 1 6`. For that output, the first iteration has a cost of 6, the last one has a cost of 2, and all others have a cost of 1.

In Sample Case #4, Reversort will necessarily perform 6 iterations, each of which will have a cost of at least 1, so there is no way the total cost can be as low as required.

Code Jam 2021 - Qualification Round

Median Sort

Problem

You want to sort $\mathbf{N}$ distinct items, $x_1, x_2, \ldots, x_{\mathbf{N}}$. Unfortunately, you do not have a way of comparing two of these items. You only have a way to, given three of them, find out which one is the median, that is, which one is neither the minimum nor the maximum among the three.

For example, suppose $\mathbf{N} = 5$ and you know that:

- x_1 is the median of $\{x_1, x_2, x_3\}$
- x_2 is the median of $\{x_2, x_3, x_4\}$
- x_3 is the median of $\{x_3, x_4, x_5\}$

Then, it is guaranteed that the sorted order of the elements is either x_4, x_2, x_1, x_3, x_5 or its reverse x_5, x_3, x_1, x_2, x_4. Notice that by knowing only medians, it is impossible to distinguish the order of any list from its reverse, since the median question has the same result for any three elements in both cases.

Your program will have to find the order of $\mathbf{T}$ lists of $\mathbf{N}$ elements using at most $\mathbf{Q}$ median questions in total (or $\mathbf{Q}/\mathbf{T}$ queries per list on average). In each case, finding either the right order or its reverse is considered correct. The order for each case is generated uniformly at random from all possible orders, and independently of any other information.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, the judge will send you a single line containing three integers $\mathbf{T}$, $\mathbf{N}$, and $\mathbf{Q}$: the number of test cases, the number of elements to sort within each test case, and the total number of questions you are allowed across all test cases, respectively. Then, you must process $\mathbf{T}$ test cases. Each test case consists of a series of question exchanges plus an additional exchange to provide the answer.

For a question exchange, your program must print a single line containing three distinct integers i, j, k all between 1 and $\mathbf{N}$, inclusive, which corresponds to asking the judge *"which element is the median of the set $\{x_i, x_j, x_k\}$?"* The judge will respond with a single line containing a single integer $\mathbf{L}$, meaning that the median of that set is $x_{\mathbf{L}}$ ($\mathbf{L}$ is always equal to one of i, j, or k). If you try to perform a $(\mathbf{Q}+1)$-th question exchange, the judge will simply output `-1`.

Once you are ready to state the result, print a line containing $\mathbf{N}$ integers representing the indices of the elements in sorted or reverse sorted order. The judge will respond with a single integer `1` if your answer is correct or `-1` if it is not. After receiving the judge's answer for the $\mathbf{T}$-th case, your program must finish in time in order to not receive a Time Limit Exceeded error. In addition, if you print additional information after receiving the result for the $\mathbf{T}$-th case, you will get a Wrong Answer judgment.

If the judge receives an invalidly formatted line or invalid values from your program at any moment, the judge will print a single number `-1`. After the judge prints `-1` for any of the reasons explained above, it will not print any further output. If your program continues to wait for the judge after receiving a `-1`, your program will time out, resulting in a Time Limit Exceeded error.

Notice that it is your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit Exceeded error. As usual, if the memory limit is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

Limits

Time limit: 40 seconds.
Memory limit: 1 GB.
$\mathbf{T} = 100$.

Test Set 1 (Visible Verdict)

$\mathbf{N} = 10$.
$\mathbf{Q} = 300 \cdot \mathbf{T}$.

Test Set 2 (Visible Verdict)

$\mathbf{N} = 50$.
$\mathbf{Q} = 300 \cdot \mathbf{T}$.

Test Set 3 (Hidden Verdict)

$\mathbf{N} = 50$.
$\mathbf{Q} = 170 \cdot \mathbf{T}$.

Testing Tool

You can use this testing tool to test locally or on our platform. To test locally, you will need to run the tool in parallel with your code; you can use our interactive runner for that. For more information, read the instructions in comments in that file, and also check out the Interactive Problems section of the FAQ.

Instructions for the testing tool are included in comments within the tool. We encourage you to add your own test cases. Please be advised that although the testing tool is intended to simulate the judging system, it is **NOT** the real judging system and might behave differently. If your code passes the testing tool but fails the real judge, please check the Coding section of the FAQ to make sure that you are using the same compiler as us.

Download testing tool

Sample Interaction

Judge	Solution
	Constraints
`2 5 600`	
Judge provides $\mathbf{T}$, $\mathbf{N}$, $\mathbf{Q}$	
	Case 1
	`1 2 3`
	Solution asks for the median of $\{x_1, x_2, x_3\}$
`2`	
Judge responds that the median is x_2	

```
4 2 3
```
Solution asks for the median of $\{x_4, x_2, x_3\}$

```
3
```
Judge responds that the median is x_3

```
5 4 3
```
Solution asks for the median of $\{x_5, x_4, x_3\}$

```
4
```
Judge responds that the median is x_4

```
5 4 3 2 1
```
Solution outputs the sorted list

```
1
```
Judge confirms that the answer is correct

Case 2

```
1 2 3
```
Solution asks for the median of $\{x_1, x_2, x_3\}$

```
3
```
Judge responds that the median is x_3

```
2 3 4
```
Solution asks for the median of $\{x_2, x_3, x_4\}$

```
4
```
Judge responds that the median is x_4

```
3 4 5
```
Solution asks for the median of $\{x_3, x_4, x_5\}$

```
5
```
Judge responds that the median is x_5

```
1 3 5 4 2
```
Solution outputs the sorted list

```
1
```
Judge confirms that the answer is correct

Code Jam 2021 - Qualification Round

Cheating Detection

Problem

100 players are competing in a 10000-question trivia tournament; the players are numbered from 1 to 100. Player i has a skill level of S_i and question j has a difficulty level of Q_j. Each skill level and each question difficulty are chosen uniformly at random from the range $[-3.00, 3.00]$, and independently of all other choices. For example, a player can have a skill level of 2.47853 and a question can have a difficulty level of -1.4172.

When player i tries to answer question j, the probability that they answer it correctly is $f(S_i - Q_j)$, where f is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

where e is Euler's number (approximately 2.718...), the mathematical constant. Notice that $0 < f(x) < 1$ for all x, so $f(S_i - Q_j)$ is always a valid probability. Each of these answer attempts is chosen at random independently of all other choices.

There is one exception: exactly one of the players is a cheater! The cheater is chosen uniformly at random from among all players, and independently of all other choices. The cheater behaves as follows: before answering each question, they flip a fair coin. If it comes up heads, they do not cheat and the rules work as normal. If it comes up tails, they secretly look up the answer on the Internet and answer the question correctly. Formally, they decide whether to cheat at random with 0.5 probability for each question, independently of all other choices.

The results of a tournament consist of only the per-question results (correct or incorrect) for each player. Apart from the general description above, you do not know anything about the skill levels of the players or the difficulties of the questions.

You must correctly identify the cheater in at least $\mathbf{P}$ percent of the test cases. That is, you must succeed in at least $\mathbf{P} \cdot \mathbf{T}/100$ out of $\mathbf{T}$ cases.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. The second line of the input gives the percentage of test cases, $\mathbf{P}$, that you must answer correctly for your solution to be considered correct. $\mathbf{T}$ test cases follow. Each case consists of 100 lines of 10000 characters each. The j-th character on the i-th line is `1` if the i-th player answered the j-th question correctly, or `0` if they answered it incorrectly.

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 number of the cheater (with player numbers starting from 1).

Limits

Time limit: 60 seconds.
Memory limit: 1 GB.

$\mathbf{T} = 50$.

Test Set 1 (Visible Verdict)

$\mathbf{P} = 10$.

Test Set 2 (Visible Verdict)

$\mathbf{P} = 86$.

Sample

Sample Input	Sample Output
1 0 00111010001010100000100000010001 -------------------------------- 99 lines of input omitted. Use the download button above to view the full sample input. --------------------------------	Case #1: 59

Notice that the sample input uses $\mathbf{T} = 1$ and $\mathbf{P} = 0$ and therefore does not meet the limits of any test set. The sample output for it is the actual cheater.