1. Equal Distribution

Introduction

IT Challenge Zoo has recently approached you to help them with a number of problems they are experiencing, for which they require your expertise in algorithms.

At the zoo there are N equally-sized orangutan cages, numbered 1 to N, with cage i housing K_i orangutans. Ideally there should be the same number of orangutans in each cage, so that the orangutans all have sufficient space.

The zookeepers have been tasked with redistributing the orangutans equally among the cages. However, since orangutans do not enjoy being moved, they would like to accomplish this while moving as few orangutans as possible.

Task

Given the initial distribution of orangutans among the cages, the zookeepers would like to determine the minimum number of orangutans which need to be moved in order to have the same number of orangutans in each cage. If it is not possible to have the orangutans distributed equally, they must take note of this so that they can report it to the manager.

The zookeepers have already managed to implement a simple program to deal with the input and output of this problem. You must help them by modifying their code to implement a function which determines whether or not it is possible to distribute the orangutans equally, and if so, calculates the minimum number of orangutans that need to be moved.

You can download the source code for their program, in your preferred language, from the *Resources* tab in Abacus.

Note: *you may instead write a program from scratch, but if you have not taken part in the IT Challenge or a similar contest before, you are advised to use the example code as it will correctly handle the details of input and output. Do not add any input or output statements to the program as doing so will prevent the automated marker from marking it correctly.*

Example

Suppose $N = 4$ and the cages initially contain 4, 6, 3 and 7 orangutans respectively. The orangutans can be distributed equally by moving 1 orangutan from cage 2 to cage 1, and moving 2 orangutans from cage 4 to cage 3, thus moving a total of 3 orangutans. This redistribution is optimal.

Suppose $N = 2$ and the cages initially contain 2 and 3 orangutans respectively. In this case, there is no way to distribute the orangutans equally among the cages.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case consists of a line containing the integer N, followed by a line containing the N space-separated integers $K_1, \ldots, K_N$.

Sample input

```
4
3
2 8 2
5
4 3 9 2 7
4
8 1 6 2
7
8 2 5 12 6 0 2
```

Output

For each test case, output a line of the form

```
Case #X: M
```

where X is the test case number, starting from 1, and M is the minimum number of orangutans that have to be moved in order to distribute the orangutans equally, if possible. If it is not possible to distribute the orangutans equally, then M should be the string "IMPOSSIBLE".

Sample output

```
Case #1: 4
Case #2: 6
Case #3: IMPOSSIBLE
Case #4: 11
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq N \leq 50$
- $0 \leq K_i \leq 100$

Time limit

2 seconds

2a. Arranging Bowls

Standard Bank

Introduction

The zoo owns N animal cages, arranged in a line. Every day a new bowl of food is placed in front of each animal cage. The animals in each cage eat a different type of food, so it is important to match the correct food bowl with the correct cage. The bowls are uniquely numbered from 1 to N and need to be sorted in descending order to feed the appropriate animals. Unfortunately, the worker that fills the bowls with food places the bowls in front of the cages in a random order.

Luckily, there is a zoo worker that can walk along the row of bowls and correct the order. When the worker is between two bowls, she can swap those bowls. The worker starts at the left bowl and walks towards the right — she does not change direction. If the bowls are still not sorted when she reaches the end of the line, she catches a lift back to the start and makes another walk along the line swapping bowls, and repeats this until the bowls are in the correct order.

Task

Your task is to determine the minimum number of times the zoo worker needs to walk along the line in order to ensure that the bowls are arranged in decreasing order.

Example

Suppose the starting configuration is 4, 3, 2, 1: then no walk is needed because the bowls are already sorted in decreasing order.

Suppose the starting configuration is 8, 6, 7, 5, 4, 2, 3, 1: then only one walk is required. The worker can swap the 6 and 7, which results in a sequence of 8, 7, 6, 5, 4, 2, 3, 1. Thereafter, 2 and 3 can be swapped within the same walk, in order to complete the arrangement.

Suppose the starting configuration is 2, 1, 4, 3: then within the first walk the 1 and 4 will be swapped, and then the 1 and 3. The configuration after the first walk would be 2, 4, 3, 1. Another walk is needed that will swap the 2 with adjacent bowls until it reaches the destination between the 3 and 1. Therefore, the minimum number of walks will be two in this case.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case starts with a line containing N, the number of bowls. The next line contains N space-separated integers, which are the numbers on the bowls from left to right.

Sample input

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

Output

For each test case, output a line of the form

```
Case #X: A
```

where X is the test case number, starting from 1, and A is the minimum number of walks that must be used to sort the food bowls in decreasing order.

Sample output

```
Case #1: 0
Case #2: 1
Case #3: 2
Case #4: 6
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq N \leq 100$

Time limit

2 seconds

2b. Arranging Bowls

Introduction

The zoo recently received additional funding and used the funds to expand tremendously. The zoo now owns many more animal cages. All the animals still need to be fed optimally by arranging the bowls in decreasing order. However, it will be a more challenging task to determine the minimum number of walks the zoo worker requires to arrange the bowls.

Task

Your task remains exactly the same. The input and output formats also remain unchanged.

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq N \leq 100\,000$

Time limit

2 seconds

3. Pedantic Penguin

Standard Bank

Introduction

Patsy the pedantic penguin is a popular attraction at the zoo, due to her peculiar eating ritual (which differs depending on the day of the month, K). At dinner time, one of the zookeepers hands her N fish, numbered 1 to N. Rather than eat the fish immediately, she first places them clockwise in a circle, in ascending order. After she has placed all N fish, she waddles over to fish 1, and proceeds to consume the fish by eating every Kth fish in the following manner.

Starting at fish 1, and counting clockwise around the circle, she counts down $K-1$ fish, which she doesn't eat. When she counts the Kth fish, she eats it. She then counts down another $K-1$ fish, once again not eating them, and then she eats the next fish she counts (that is, the $2K$th fish she has counted so far), and so on. When she reaches the end of the circle, she continues counting from the beginning of the circle, and when she reaches the position of a fish which she has already eaten, she skips over that position without counting it. This process continues until there is only one fish left in the circle, and she finishes off her meal by eating that fish.

The zookeeper would like to administer some medication to Patsy, which she should not have on an empty stomach. He has decided to put the medication inside the last fish she will eat, so that she will only ingest the medication at the end of her meal.

Task

Help the zookeeper with the task of figuring out which fish Patsy will eat last.

Example

Suppose $N = 9$ and $K = 2$. Then Patsy will count down the fish as follows, with an underline indicating that she eats the fish: 1, $\underline{2}$, 3, $\underline{4}$, 5, $\underline{6}$, 7, $\underline{8}$, 9, $\underline{1}$, 3, $\underline{5}$, 7, $\underline{9}$, 3, $\underline{7}$, 3, $\underline{3}$. Thus, she will eat fish 3 last.

Suppose $N = 4$ and $K = 5$. Then she will count down the fish as follows: 1, 2, 3, 4, $\underline{1}$, 2, 3, 4, 2, $\underline{3}$, 4, 2, 4, 2, $\underline{4}$, 2, 2, 2, 2, $\underline{2}$. So, the medication should be put inside fish 2.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case consists of a single line containing the space-separated integers N and K.

Sample input

```
5
9 2
4 5
18 1
11 5
41 3
```

Output

For each test case, output a line of the form

```
Case #X: F
```

where X is the test case number, starting from 1, and F is the number of the last fish Patsy eats.

Sample output

```
Case #1: 3
Case #2: 2
Case #3: 18
Case #4: 8
Case #5: 31
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \le T \le 20$
- $1 \le N \le 10^{12}$
- $1 \le K \le 31$

Time limit

2 seconds

4. Pigeon Sorting

Standard Bank

Introduction

The zoo owns N pigeons and N pigeon-holes, both numbered from 1 to N. Each pigeon is supposed to go into the hole with the matching number, but they have become mixed up and need to be put back into the correct holes.

The zoo has purchased a new pigeon-sorting machine. Unfortunately, budget cuts meant that they had to get a low-end model that has limited features and has to be programmed manually. The machine is programmed by entering a list of K numbers $a_1, a_2, \ldots, a_K$ into the machine, where no number may appear more than once in the list. Once the list has been entered, the operator presses **Start** and moves quickly out of the way. The machine then moves the pigeon in hole a_i to hole a_{i+1} (for $1 \leq i < K$) and the pigeon in hole a_K to hole a_1.

Task

Every time the **Start** button is pressed there is a risk that the operator will get caught in the machine and have his/her internal organs sorted, so the zoo wants to minimise the number of times the machine is used. Help them by producing a minimal number of lists with which to program the machine. Note that the lengths of the lists do not matter.

Example

Suppose that holes 1–5 are currently occupied by pigeons 2, 1, 3, 5, 4 respectively. There is no way to put the pigeons in the correct holes using the machine only once, but several ways to do it by using the machine twice. One such way is as follows:

- Program the machine with 1, 2, 4, 5. After this the holes will contain pigeons 4, 2, 3, 1, 5.
- Program the machine with 4, 1. After this the holes will contain pigeons 1, 2, 3, 4, 5.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case starts with a line containing N, the number of pigeons. The next line contains N space-separated integers, the ith of which is the number of the pigeon in hole i.

Sample input

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

Output

For each test case, output a line of the form

```
Case #X: A
```

where X is the test case number, starting from 1, and A is the minimum number of times the machine must be used. This must be followed by A lines of space-separated integers, each of which is one list given to the machine, in the order they should be executed.

If there are multiple valid solutions, you may output any of them.

Sample output

```
Case #1: 2
1 2 4 5
4 1
Case #2: 1
1 2 3 4 5
Case #3: 0
Case #4: 2
1 4 6 2 8 3 7
3 2 1
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$
- $1 \leq N \leq 1\,000$
- Each pigeon number from 1 to N will appear exactly once in a test case.

Time limit

2 seconds

5. Dice Rolling

Introduction

There are many jobs in a zoo, and some of them can be unpleasant or dangerous: cleaning up after the elephants, brushing the tigers' teeth, putting in the rhinos' contact lenses.... Every month the zookeepers play a game using dice to decide who gets the nice jobs and who gets the nasty ones.

The game is played using N identical dice, each with M sides numbered 1 to M. The M sides are all equally likely to be rolled. When it is a player's turn, he takes the following steps:

- He rolls all N dice.
- He picks up some (possibly all or none) of the dice, and rolls them again.
- He determines how many points he has scored, based on the N faces showing.

Particular combinations of dice are worth points, but the combinations and the number of points they are worth change from month to month. For example, suppose that this month one scores 3 points for having three 4's, 2 points for having a 4, a 5 and a 6, and 1 point for having a 1, 2 and 3. If the final result after re-rolling is 4, 6, 5, 4, 1, 4, then you will score 5 points. Note that one of the 4's is used both to make three 4's and to make 4–5–6; this is allowed. Also note that the order of the player's dice does not matter. On the other hand, rolling six 4's only scores 3 points, not 6, because the points for three 4's can only be earned once.

Task

One of the zookeepers has asked you to help him do better in this game. Given his initial roll and this month's combinations, he wants to select the dice to re-roll that will maximise his expected (i.e., average) score. Help him by determining this maximum expected score.

Example

Suppose that $N = 6$, $M = 6$ and the points are awarded as in the introduction. If the zookeeper rolls 1, 4, 4, 3, 6, 2, then he should re-roll the 1, 2 and 3. There is a 2.8% chance of scoring 1 point, a 28.2% chance of scoring 2 points, a 28.2% chance of scoring 3 points, and a 13.9% chance of scoring 5 points (these percentages are approximate). Thus, on average, the zookeeper can expect to get 2.134 points, and this is the best possible.

Input

The input contains multiple test cases. The first line contains T, the number of test cases. Each test case starts with a line containing space-separated integers N, M and P, where P is the number of different combinations that score points. The next line contains N space-separated integers, the initial roll.

The following P lines each describe one way to score points. The ith of these lines contains M integers $r_{i,1}$ to $r_{i,M}$, and an integer S_i, all separated by single spaces. If a player finishes with at least $r_{i,j}$ dice with side j for all $1 \leq j \leq M$, then he scores S_i points.

Sample input

```
3
2 5 1
4 2
0 0 0 0 1 10
6 6 3
1 4 4 3 6 2
0 0 0 3 0 0 3
0 0 0 1 1 1 2
1 1 1 0 0 0 1
6 6 2
1 2 3 4 4 4
1 1 1 1 1 1 10
1 1 1 1 1 1 5
```

Output

For each test case, output a line of the form

```
Case #X: E
```

where X is the test case number, starting from 1, and E is the maximum possible expected (average) score. E must be output with exactly 3 decimal places: refer to the contest manual for guidelines on how to do this in each language.

Sample output

```
Case #1: 3.600
Case #2: 2.134
Case #3: 0.833
```

Constraints

It is guaranteed that in the input used to test your program:

- $1 \leq T \leq 20$

- $1 \leq N \leq 6$
- $2 \leq M \leq 6$
- $1 \leq P \leq 10$
- $1 \leq S_i \leq 100$
- $0 \leq r_{i,j} \leq N$
- $r_{i,1} + r_{i,2} + \cdots + r_{i,M} \leq N$

Time limit

10 seconds