Practice Round APAC test 2017

A. Lazy Spelling Bee

B. Robot Rock Band

C. Not So Random

D. Sums of Sums

Questions asked

- Submissions

Lazy Spelling Bee

5pt | Not attempted
698/1266 users
correct (55%)

8pt | Not attempted
496/685 users
correct (72%)

Robot Rock Band

6pt | Not attempted
480/622 users
correct (77%)

14pt | Not attempted
142/407 users
correct (35%)

Not So Random

11pt | Not attempted
204/310 users
correct (66%)

20pt | Not attempted
109/158 users
correct (69%)

Sums of Sums

8pt | Not attempted
230/395 users
correct (58%)

28pt | Not attempted
13/128 users
correct (10%)

- Top Scores

Jayam	100
Seter	100
KillswitcherEngag...	100
onepunchman	100
Sumeet.Varma	100
gdragon007	100
libenchao	100
jpravishAA	100
vaibhav227	100
wrong	100

Problem A. Lazy Spelling Bee

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
5 points

[Solve A-small]

Large input
8 points

[Solve A-large]

Problem

In the Lazy Spelling Bee, a contestant is given a target word W to spell. The contestant's answer word A is acceptable if it is the same length as the target word, and the i-th letter of A is either the i-th, (i-1)th, or (i+1)th letter of W, for all i in the range of the length of A. (The first letter of A must match either the first or second letter of W, since the 0th letter of W doesn't exist. Similarly, the last letter of A must match either the last or next-to-last letter of W.) Note that the target word itself is always an acceptable answer word.

You are preparing a Lazy Spelling Bee, and you have been asked to determine, for each target word, how many distinct acceptable answer words there are. Since this number may be very large, please output it modulo 1000000007 ($10^9 + 7$).

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of one line with a string consisting only of lowercase English letters (a through z).

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 distinct acceptable answer words, modulo $10^9 + 7$.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ length of each string ≤ 5.

Large dataset

1 ≤ length of each string ≤ 1000.

Sample

```
Input       Output

4           Case #1: 4
ag          Case #2: 1
aa          Case #3: 108
abcde       Case #4: 1
x
```

In sample case #1, the acceptable answer words are aa, ag, ga, and gg.

In sample case #2, the only acceptable answer word is aa.

Powered by

Google Cloud Platform

Problem B. Robot Rock Band

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
14 points | Solve B-large |

Problem

You're the manager of Xorbitant, the world's first robot rock band. There will be four positions in the band, and there are **N** robots auditioning for each position. (No robot is auditioning for more than one position.) Every robot has a number, and multiple robots might have the same number, just as two people can have the same name.

You know from market research that your robot audiences won't care how well the robot band members make music, how handsome they are, or what scandalous things the tabloids say about them. Instead, the audience will be checking to see whether the four members' numbers, when bitwise XORed together, equal a certain trendy number **K**.

How many different sets of four robots (one for each position) is it possible to choose so that the band will have this property? More specifically, given four lists A, B, C, D containing **N** numbers each, how many ways are there to choose one number a from list A, one number b from list B, and so on, such that a^b^c^d = **K**? (Here ^ represents the bitwise XOR operation.)

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case begins with one line with two space-separated integers, **N** and **K**, as described above. Then, four more lines follow. Each has **N** space-separated integers and represents the ID numbers of the robots auditioning for a certain position in the band.

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 different bands that meet the conditions.

Limits

$1 \le$ **T** ≤ 10.
$0 \le$ **K** $\le 10^9$.
$0 \le$ all robot numbers $\le 10^9$.

Small dataset

$1 \le$ **N** ≤ 50.

Large dataset

$1 \le$ **N** ≤ 1000.

Sample

```
Input        Output

2            Case #1: 4
2 3          Case #2: 8
0 0
2 0
0 0
0 1
2 0
1 10
1 10
1 10
1 10
```

In sample case #1, in order to get a combined bitwise XOR of 3, the robot chosen from the second list must be 2, and the robot chosen from the fourth list must be 1. For the first and third lists, either of the two 0 robots can be chosen, so there are 2 * 2 = 4 possible bands that meet the criteria. Note that

even though all of these bands are of the form (0, 2, 0, 1), they are considered
different because the selections from the lists were different.

Powered by

Google Cloud Platform

Problem C. **Not So Random**

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 11 points | Solve C-small |
| Large input 20 points | Solve C-large |

Problem

There is a certain "random number generator" (RNG) which takes one nonnegative integer as input and generates another nonnegative integer as output. But you know that the RNG is really not very random at all! It uses a fixed number **K**, and always performs one of the following three operations:

- with probability **A**/100: return the bitwise AND of the input and **K**
- with probability **B**/100: return the bitwise OR of the input and **K**
- with probability **C**/100: return the bitwise XOR of the input and **K**

(You may assume that the RNG *is* truly random in the way that it chooses the operation each time, based on the values of **A**, **B**, and **C**.)

You have **N** copies of this RNG, and you have arranged them in series such that output from one machine will be the input for the next machine in the series. If you provide **X** as an input to the first machine, what will be the expected value of the output of the final machine in the series?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of one line with six integers **N**, **X**, **K**, **A**, **B**, and **C**. Respectively, these denote the number of machines, the initial input, the fixed number with which all the bitwise operations will be performed (on every machine), and 100 times the probabilities of the bitwise AND, OR, and XOR operations.

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 expected value of the final output. y will be considered correct if it is within an absolute or relative error of 10^{-9} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

$1 \le$ **T** ≤ 50.
$0 \le$ **A** ≤ 100.
$0 \le$ **B** ≤ 100.
$0 \le$ **C** ≤ 100.
A+B+C = 100.

Small dataset

$1 \le$ **N** ≤ 10.
$0 \le$ **X** $\le 10^4$.
$0 \le$ **K** $\le 10^4$.

Large dataset

$1 \le$ **N** $\le 10^5$.
$0 \le$ **X** $\le 10^9$.
$0 \le$ **K** $\le 10^9$.

Sample

Input	Output
3	Case #1: 3.0000000000
1 5 5 10 50 40	Case #2: 3.6000000000
2 5 5 10 50 40	Case #3: 15.6850579098
10 15 21 70 20 10	

In sample test case #1, the final output will be 5 if AND or OR happens and 0 if XOR happens. So the probability of getting 5 is (0.1 + 0.5) and the probability

of getting 0 is 0.4. So the expected final output is 5 * 0.6 + 0 * 0.4 = 3.

In sample test case #2, the final output will be 5 with probability 0.72, and 0 otherwise.

Problem D. Sums of Sums

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
8 points

[Solve D-small]

Large input
28 points

[Solve D-large]

Problem

Alice presented her friend Bob with an array of **N** positive integers, indexed from 1 to **N**. She challenged Bob with many queries of the form "what is the sum of the numbers between these two indexes?" But Bob was able to solve the problem too easily.

Alice took her array and found all **N***(**N**+1)/2 non-empty subarrays of it. She found the sum of each subarray, and then sorted those values (in nondecreasing order) to create a new array, indexed from 1 to **N***(**N**+1)/2. For example, for an initial array [2, 3, 2], Alice would generate the subarrays [2], [3], [2], [2, 3], [3, 2], and [2, 3, 2] (note that [2, 2], for example, is **NOT** a subarray). Then she'd take the sums -- 2, 3, 2, 5, 5, 7 -- and sort them to get a new array of [2, 2, 3, 5, 5, 7].

Alice has given the initial array to Bob, along with **Q** queries of the form "what is the sum of the numbers from index L_i to R_i, inclusive, in the new array?" Now Bob's in trouble! Can you help him out?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with one line with two space-separated integers **N** and **Q**, denoting the number of elements in the initial array and the number of Alice's queries. Then, there is one line with **N** space-separated integers, denoting the elements of Alice's initial array. Finally, there are **Q** more lines with two space-separated integers each: L_i and R_i, the inclusive index bounds for the i-th query.

Output

For each test case, output one line with `Case #x:`, where **x** is the test case number (starting from 1). Then output **Q** more lines, each with one integer, representing the answers to the queries (in the order they were asked).

Limits

1 ≤ **T** ≤ 10.
1 ≤ **Q** ≤ 20.
1 ≤ each element of the initial array ≤ 100.
1 ≤ L_i ≤ R_i ≤ N*(N+1)/2.

Small dataset

1 ≤ **N** ≤ 10^3.

Large dataset

1 ≤ **N** ≤ 200000.

Sample

```
Input           Output

1               Case #1:
5 5             1
5 4 3 2 1       45
1 1             105
1 10            26
1 15            48
3 8
4 11
```

In sample case #1, Alice's new array would be: [1, 2, 3, 3, 4, 5, 5, 6, 7, 9, 9, 10, 12, 14, 15].