Common Palindrome

A **palindrome** is a string that reads the same from the left as it does from the right. Given two strings **A** and **B**, you need to find the length of longest palindrome which is a subsequence of both **A** and **B**. A subsequence is a sequence obtained by deleting zero or more characters from a string. For example, say, **A** = "cfcfaafc", **B** = "efagfc". Then the longest palindrome which is a subsequence of both A and B is "faf". So the answer is 3.

Input

First line of the input contains a positive integer **T** (**T <= 100**). Each of the following **T** cases consists of 2 lines. These 2 lines contain the strings **A** and **B**, respectively. Length of **A** and **B** will not be more than **60**. All these strings contain only lowercase letters **('a' -'z')**. No empty strings will appear in the input.

Output

For each case, print a line of the form **Case <x> : <y>**, where **x** is the case number and **y** is the length of the longest common palindromic subsequence.

Sample Input	Sample Output
3 cfcfaafc Efagfc Afbcdfca Bcadfcgyfka Palin drome	Case 1: 3 Case 2: 5 Case 3: 0

Just Make A Wish

If you are given two sides of a rectangle, you can find the area of the rectangle very easily. But can you do the opposite? Given the area of a rectangle can you tell me how many different rectangles can have this area, provided the sides of the rectangles must be integer? Two rectangles are equal only if the sides are same. You cannot rotate a rectangle. So if the area is 6 there are 4 different rectangles are possible. They are 1x6, 2x3, 3x2 and 6x1.

This problem is very easy and I am sure you can solve it in a minute. Your problem is much tougher. One of your friends Po has got a magic Genie. Every day the Genie grants Po a wish. When asked to grant the wish, the Genie (depending on its mood) tells a number. If it's positive then Po's land gets multiplied by that number. If it's negative then Po's land is divided by the absolute value of that number. So every day, Po tells you the area of the land Po owns. You need to find out **how many different rectangles** can have that area. And then output the summation after *D* days. Initially the area of land Po owns **is 1**.

Input

First line of input consists of an integer, T ($T \leq 10$), the number of test cases. Each case starts with an integer, D ($0 < D \leq 10^6$), the number of days this wish granting will go on. Next D lines each has an integer G ($0 < |G| \leq 10^6$), the number with which Po's lands area will be multiplied or divided. If G is negative, it will be a divisor of Po's land area.

Output

For each case print one line `Case x:' where X is the case number. Then for each wish of a day, find the number of different of rectangles you can make which will have the same area as Po's land and output the summation after D days. As this can be much bigger, output modulo 1000000007 ($10^9 + 7$).

Sample Explanation:

1st day: Area = 12 $\rightarrow$ 6 ways.

2nd day: Area = 36 $\rightarrow$ 9 ways.

3 day: Area= 18 $\rightarrow$ 6 ways.

4 day: Area = 90 $\rightarrow$ 12 ways.

5 day: Area = 15 $\rightarrow$ 4 ways.

So in total (6+9+6+12+4) = 37 ways.

Sample Input

```
1
5
12
3
-2
5
-6
```

Sample Output

```
Case 1: 37
```

Ahoy, Pirates!

Input: Standard Input
Output: Standard Output

In the ancient pirate ages, the Pirate Land was divided into two teams of pirates, namely, the Buccaneer and the Barbary pirates. Each pirate's team was not fixed, sometimes the opponent pirates attacked and he was taken away to the other pirate team. All on a sudden a magician appeared in the Pirate Land, where he was making transition of pirates from their team to other team at his own will. Of course, handy spells were used. The process of changing team was known as mutating.

There were N pirates and all of the pirates have a unique id from 0 to N-1. The great magician could mutate a bunch of pirates with consecutive id's to another one.

Suppose there were 100 pirates in the pirate land and all of them were Barbary pirates, then the magician could cast a spell to change pirates with id's from 10 to 33 to Buccaneer pirates. Then the whole pirate land would have 24 Buccaneer and 76 Barbary pirates.

The magician was very fast casting the spell. Once, God started to dislike this. God had favor for the Buccaneer pirates and God asked the magician, "Tell me, how many of the pirates of index from 2 to 30 are Buccaneer pirates?". Now the magician was puzzled as he was only efficient in casting spells, not in counting ☺

Being clever enough, the magician captured a clever man from the Earth Land. And unfortunately it's you! Now you have to answer the God's questions.

Input

The first line of input will contain number of test cases T.
For each test case:
The first part of the description will be of the pirate land. There could be up to N (1<=N<=1024000) pirates. Each pirate is either assigned to Buccaneer or Barbary Pirate. Buccaneer pirates are described by '1' (ONE) and Barbary pirates are described by '0' (ZERO). You have to build a string of the pirates' description. Each case starts with an integer M (M<=100), where M pair lines follow. In each pair of lines (we call it a set), first has an integer **T** (T <= 200) and next one has a nonempty string **Pirates** (consisting of 0 and 1, 0 for Barbary, 1 for Buccaneer, has maximum length of 50). For each pair concatenate the string **Pirates**, **T** times. Concatenate all the resulting M sets of strings to build the pirate description. The final concatenated string describes the pirates from index 0 to end (N-1 for N pirates).
Now the next part of the input will contain queries. First line of next part has an integer Q describing number of queries. Each subsequence Q (1<=Q<=1000) lines describe each query. Each query has a string F or E or I or S and two integers, a and b denoting indexes. The meaning of the query string are follows:
F a b, means, mutate the pirates from index a to b to Buccaneer Pirates.
E a b, means, mutate the pirates from index a to b to Barbary Pirates.
I a b, means, mutate the pirates from index a to b to inverse pirates.
S a b, means, "God's query" God is asking a question: "Tell me, how many Buccaneer pirates are there from index a to b?"
(a <= b, 0 <= a < n, 0 <= b < n, index range are inclusive)

Output

For each test print the case number as the sample output suggests. Then for each of "God's query", output the query number, colon (:) and a space and the answer to the query as the sample suggest.

Sample Input	**Output for Sample Input**
2 2 5 10 2 1000 5 F 0 17 I 0 5 S 1 10 E 4 9 S 2 10	Case 1: Q1: 5 Q2: 1 Case 2: Q1: 0

3 3 1 4 0 2 0 2 I 0 2 S 0 8	

Explanation:
Case1:
The pirate land is as follows (N = 18)
101010101010001000
Before God's first query it was as follows
000000111111111111

Case 2:
The pirate land is as follows (N=9)
111000000

Subtraction Game 1

Chef is playing a game on a sequence of **N** positive integers, say $\mathbf{A_1, A_2, \ldots A_N}$. The game is played as follows.

- If all the numbers are equal, the game ends.
- Otherwise
- Select two numbers which are unequal
- Subtract the smaller number from the larger number
- Replace the larger number with the result from above (see the explanation section for clarity)

Chef has already figured out that the game **always terminates**. He also knows, for a given sequence of integers, the game will always terminate on the **same value**, no matter how the game is played. Chef wants you to simulate the game for him and tell him on which value will the game terminate for a given sequence of integers.

Input

The first line of the input contains an integer **T**, the number of test cases. Then follow the description of **T** test cases. The first line of each test case contains a single integer **N**, the length of the sequence. The second line contains **N** positive integers, each separated by a single space.

Output

For each test case, **output a single integer** - the value of all the numbers when they are **equal** (and the game terminates), on a line by itself.

Constraints

$1 \leq T \leq 100$
$1 \leq N \leq 1000$
$1 \leq A_i \leq 10^9$

Input

3
2
10 12
2
5 9
3
6 10 15

Output

2
1
1

Explanation

Test Case 1: Since there are only two numbers, the operations are forced.

- { 10, 12 } => Replace 12 with (12 - 10 = 2) => { 10, 2 }
- { 10, 2 } => Replace 10 with (10 - 2 = 8) => { 8, 2 }
- { 8, 2 } => Replace 8 with (8 - 2 = 6) => { 6, 2 }
- { 6, 2 } => Replace 6 with (6 - 2 = 4) => { 4, 2 }
- { 4, 2 } => Replace 4 with (4 - 2 = 2) => { 2, 2 }

The value of all the numbers when the game ends is **2**.

Test Case 2: Since there are only two numbers, the operations are forced.

- { 5, 9 } => Replace 9 with (9 - 5 = 4) => { 5, 4 }
- { 5, 4 } => Replace 5 with (5 - 4 = 1) => { 1, 4 }
- { 1, 4 } => Replace 4 with (4 - 1 = 3) => { 1, 3 }
- { 1, 3 } => Replace 3 with (3 - 1 = 2) => { 1, 2 }

- { 1, 2 } => Replace 2 with (2 - 1 = 1) => { 1, 1 }

The value of all the numbers when the game ends is **1**.

Test Case 3: One way to play the game is

- { 6, 10, 15 } => Replace 15 with (15 - 6 = 9) => { 6, 10, 9 }
- { 6, 10, 9 } => Replace 10 with (10 - 6 = 4) => { 6, 4, 9 }
- { 6, 4, 9 } => Replace 9 with (9 - 6 = 3) => { 6, 4, 3 }
- { 6, 4, 3 } => Replace 6 with (6 - 4 = 2) => { 2, 4, 3 }
- { 2, 4, 3 } => Replace 3 with (3 - 2 = 1) => { 2, 4, 1 }
- { 2, 4, 1 } => Replace 4 with (4 - 2 = 2) => { 2, 2, 1 }
- { 2, 2, 1 } => Replace first 2 with (2 - 1 = 1) => { 1, 2, 1 }
- { 1, 2, 1 } => Replace 2 with (2 - 1 = 1) => { 1, 1, 1 }

The value of all the numbers when the game ends is **1**. You may try to play the game differently and observe that **the game will always end when all the values are 1**.

Subtraction Game 2

Chef is playing a game on a sequence of **N** positive integers, say $\mathbf{A_1, A_2, \ldots A_N}$. The game is played as follows.

- If all the numbers are equal, the game ends.
- Otherwise
- Select two numbers which are unequal
- Subtract the smaller number from the larger number
- Replace the larger number with the result from above

Chef has already figured out that the game **always terminates**. He also knows, for a given sequence of integers, the game will always terminate on the **same value**, no matter how the game is played. Chef wants you to simulate the game for him and tell him if the game terminates on **1**.

In fact, there may be many such games. Given a sequence of integers Chef wants to know the number of **sub-sequences** of the given sequence, for which, playing the above game on the subsequences will terminate on **1**. A **sub-sequence** can be obtained from the original sequence by **deleting 0 or more** integers from the original sequence. See the explanation section for clarity.

Input

The first line of the input contains an integer **T**, the number of test cases. Then follow the description of **T** test cases. The first line of each test case contains a single integer **N**, the length of the sequence. The second line contains **N** positive integers, each separated by a single space.

Output

For each test case, **output a single integer** - the number of **sub-sequences** of the original sequence, such that, playing the game on the **sub-sequence** results in ending the game with all the values equal to**1**.

Constraints

$1 \leq T \leq 100$
$1 \leq N \leq 60$
$1 \leq A_i \leq 10^4$
All A_i will be distinct.

Input

3
4
2 3 5 7
4
3 4 8 16
3
6 10 15

Output

11
7
1

Explanation

Test Case 1: The following 11 sub-sequences are counted.

- { 2, 3 }
- { 2, 5 }
- { 2, 7 }
- { 3, 5 }
- { 3, 7 }
- { 5, 7 }
- { 2, 3, 5 }
- { 2, 3, 7 }
- { 2, 5, 7 }
- { 3, 5, 7 }

- { 2, 3, 5, 7 }

Test Case 2: The following 7 sub-sequences are counted.

- { 3, 4 }
- { 3, 8 }
- { 3, 16 }
- { 3, 4, 8 }
- { 3, 4, 16 }
- { 3, 8, 16 }
- { 3, 4, 8, 16 }

Test Case 3: There are 8 subsequences of { 6, 10, 15 }

- {} => The game cannot be played on this sub-sequence
- { 6 } => The game cannot be played on this sub-sequence
- { 10 } => The game cannot be played on this sub-sequence
- { 15 } => The game cannot be played on this sub-sequence
- { 6, 10 } => The game cannot end at { 1, 1 }
- { 6, 15 } => The game cannot end at { 1, 1 }
- { 10, 15 } => The game cannot end at { 1, 1 }
- { 6, 10, 15 } => The game ends at {1, 1, and 1}. Hence this is the only sub-sequence that is counted in the result.