1. Completing Sequences

Standard Bank

Introduction

Two common types of mathematical sequences are *arithmetic* and *geometric progressions*. In an arithmetic progression, each term is the previous one plus some integer constant, e.g., 7, 9, 11, 13 (each term is the previous one plus 2). In a geometric progression, each term is the previous one multiplied by some integer constant, e.g. 3, 12, 48, 192 (each term is the previous one times 4). Note that for the purposes of this problem, the ratio must be an integer, so 192, 48, 12, 3 is *not* considered to be a geometric progression, even though each term is a quarter of the previous one.

Task

One of your colleagues has written a program that detects *arithmetic progressions* and computes the next term. You must modify her code to instead detect geometric progressions and compute the next term.

You can download the source code for your colleague's 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

Given the sequence 3, 12, 48, 192, your program should determine that the next term is 768.

Input

The input file 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 terms given in the sequence. This is followed by a line containing N integers separated by spaces, the given terms.

Sample input

```
6
4
3 12 48 192
4
192 48 12 3
5
3 5 7 9 11
4
1 -2 4 -8
2
0 0
4
0 1 0 -1
```

Output

For each test case, output a line of the form

```
Case #X: P
```

where X is the test case number, starting from 1, and P is the next term in the geometric progression. If the sequence is not a geometric progression, then instead output

```
Case #X: NO
```

Note that the output is case sensitive.

Sample output

```
Case #1: 768
Case #2: NO
Case #3: NO
Case #4: 16
Case #5: 0
Case #6: NO
```

Constraints

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

- $1 \leq T \leq 20$
- $2 \leq N \leq 20$
- $-1\,000 \leq \text{each given term} \leq 1\,000$

Time limit

2 seconds

2. Palindromic Phrases

Introduction

Anna and Bob are playing a game to see who can think of the longest palindromic phrase. A phrase is palindromic if it reads the same backward and forward when punctuation, capitalisation, and spaces are ignored. They became very frustrated checking the sentences by hand, since it is tedious and prone to errors. They asked you to help them by writing a program that checks whether their phrases are palindromic or not.

Task

Given a set of phrases, determine which of the phrases are palindromic when punctuation, capitalisation and spaces are ignored.

Example

Suppose Bob comes up with the phrase `Madam, I am Adam`. Unfortunately, this phrase is not palindromic. However, Anna alters Bob's phrase to `Madam, I'm Adam`, which is palindromic.

Input

The first line of input contains the integer T, the number of phrases. The following T lines each contain one of these phrases.

Sample input

```
5
Madam, I am Adam
Madam, I'm Adam
Rise to vote, sir!
Is this a palindrome?
Was it a car or a cat I saw?
```

Output

For each test case, output a line of the form

```
Case #X: P
```

where X is the phrase number, starting from 1, and P is "YES" if the phrase is palindromic and "NO" if not. Note that the output format is case sensitive.

Sample output

```
Case #1: NO
Case #2: YES
Case #3: YES
Case #4: NO
Case #5: YES
```

Constraints

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

- $1 \leq T \leq 20$
- $1 \leq$ length of phrase $\leq 1\,000$
- Each phrase will only consist of the characters `a-z`, `A-Z`, space and the following punctuation: `.,!?-'"`.
- A phrase will not have any leading, trailing, or double spaces and will contain at least one letter.

Time limit

2 seconds

3. Scrambled Letters

Introduction

You are playing a word game with your friends. In each round some tiles, each bearing a single letter, are selected and placed in a row. The tiles are presented to all the players, and the player who can form the longest word using the letters in the row wins the round. If there is a tie, the player whose word occurs first in the dictionary wins.

The tiles can be rearranged to form the word and they do not all have to be used, but a tile can only be used once in a round. You are eager to win, but unfortunately you do not have a good vocabulary. You have devised a cunning plan: you are going to write a program to help you.

Task

Given a list of words in the dictionary and the row of letter tiles selected during each round of the game, find the best word that can be played in each round.

Example

Suppose the dictionary consists of the words `desserts`, `set`, `sets`, `stress` and `stressed`, in that order. The longest word that can be formed from `tsabeyz` is `set`. The word `sets` cannot be constructed since only one `s`-tile is available.

Both the word `desserts` and `stressed` can be constructed from `sdtssree`, but `desserts` occurs first in the dictionary.

Input

The first line of input contain the integer T, the number of test cases. This is followed by T test cases.

The first line of each test case contains two space-separated integers: M and N, representing the number of words in the dictionary and the number of rounds in the game, respectively.

The next M lines contains the dictionary, one word per line and listed in alphabetical order. Each word consists of lower-case letters. This is followed by N lines listing the tiles selected in each round, one line per round. Each line contains only lower-case letters, without spaces.

Sample input

```
2
5 3
desserts
set
sets
stress
stressed
tsabeyz
zebra
sdtssree
3 1
clambers
rambles
scramble
ambslrce
```

Output

For each test case, output a line of the form

```
Case #X:
```

where X is the test case number, starting from 1. Then, for each round output the best word in the dictionary that can be played in that round. If no word can be constructed from the letters, output “NONE”. Note that the output format is case-sensitive.

Sample output

```
Case #1:
set
NONE
desserts
Case #2:
clambers
```

Constraints

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

- $1 \leq T \leq 5$
- $1 \leq M \leq 2\,000$, $1 \leq N \leq 100$
- $1 \leq$ length of each word in dictionary ≤ 50
- $1 \leq$ length of each row ≤ 50
- The words in the dictionary will be unique.

Time limit

5 seconds

4. Carpet Laying

Introduction

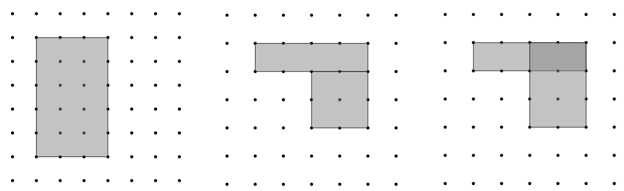
Standard Bank wants to renovate a new office building that they bought. One of the renovation tasks is to replace the old flooring with brand new carpets. The area where new carpets need to be installed can be described by multiple (possible intersecting) rectangles. Each rectangle can be described by the upper-left and lower-right corners. Let the integer coordinate of the upper-left corner be denoted as (X_1, Y_1) and the lower-right corner as (X_2, Y_2). The coordinates of the upper-left corner of the building are at (0, 0).

Task

Given the integer corner coordinates of a number of rectangles which must be carpeted, find the total area of carpeting that must be bought to cover all of them.

Example

The carpet layout of the first three sample input buildings.



Input

The input file 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 rectangles given. This is followed by N lines containing 4 integers namely X_1, Y_1, X_2 and Y_2 separated by spaces, the given rectangle corner coordinates.

Sample input

```
4
1
1 1 4 6
2
1 1 5 2
3 2 5 4
2
1 1 5 2
3 1 5 4
1
0 0 123456 789012
```

Output

For each test case, output a line of the form

```
Case #X: A
```

where X is the test case number, starting from 1; A is the total area covered by the rectangles.

Sample output

```
Case #1: 15
Case #2: 8
Case #3: 8
Case #4: 97408265472
```

Constraints

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

- $1 \leq T \leq 20$
- $1 \leq N \leq 100$
- $0 \leq X_1, Y_1, X_2, Y_2 \leq 1\,000\,000\,000$
- $X_1 \leq X_2$
- $Y_1 \leq Y_2$

Time limit

5 seconds

5. Sequence Search

Standard Bank

Introduction

An interesting binary sequence can be defined recursively as

$$\begin{aligned} t_0 &= 0 \\ t_{2n} &= t_n \\ t_{2n+1} &= 1 - t_n. \end{aligned}$$

The first few terms in the sequence are:

$$0110100110010110100101100 11.$$

Task

Your task is to determine if a given binary string ever occurs as a contiguous substring of the sequence and, if it does, to find its earliest occurrence.

Example

The string 101 first occurs at location 2, the string 1011 first occurs at location 11, but the string 111 never occurs anywhere in the sequence.

Input

The first line of input contains an integer T, the number of test cases. The next T lines each contain the binary string, S, to be found in the sequence for that test case.

Sample input

```
4
101
1011
111
10100101101
```

Output

For each test case, output a line of the form

```
Case #X: L
```

where X is the test case number, starting from 1, and L is the index of the first occurrence of the string or "NEVER" if it never occurs in the sequence. Note that the output format is case-sensitive.

Sample output

```
Case #1: 2
Case #2: 11
Case #3: NEVER
Case #4: 42
```

Constraints

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

- $1 \leq T \leq 25$
- $1 \leq$ the length of $S \leq 600\,000$

Time limit

3 seconds

6. Arranging Artefacts

Introduction

A museum curator wishes to display a selection of artefacts at an exhibition. The artefacts are cuboidal and while they are all of similar height, they have different widths and depths. The curator does not care which direction artefacts are facing and so any artefacts may be rotated by 90 degrees so that their 'widths' and 'depths' are effectively interchanged. He wants the artefacts to be arranged in a single row in order of *increasing* width, but *decreasing* depth. He does not mind if adjacent artefacts have the same widths or depths.

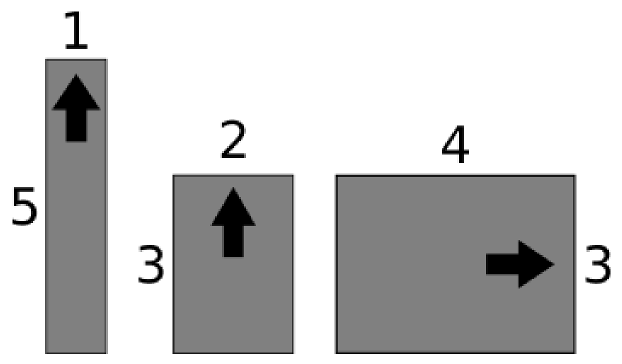
Task

You are given the widths and depths of each of the artefacts. You should determine whether it is possible for the curator to arrange them in his desired way.

Example

Suppose there are 3 artefacts. The first artefact has width 3m and depth 4m, the second has width 1m and depth 5m, and the third has width 2m and depth 3m. Suppose they are arranged as in the diagram with the second artefact on the left, the third in the middle and the first on the right and with the first artefact rotated. Then the widths will be in the order 1m, 2m, 4m and the depths will be in the order: 5m, 3m, 3m. This satisfies the constraints.

On the other hand, if there is one artefact of width 1m and depth 1m and one artefact of width 2m and depth 2m, no valid arrangement is possible.



Input

The first line of input contains an integer T, the number of test cases. The first line of each test case contains a single integer: N, the number of artefacts. The next N lines contain two space-separated positive integers: w_i and d_i, the width and depth of each artefact in metres.

Sample input

```
3
3
3 4
1 5
2 3
2
1 1
2 2
3
1 3
2 4
3 5
```

Output

For each test case, output a line of the form

```
Case #X: P
```

where X is the test case number, starting from 1, and P is "YES" if it is possible to arrange the artefacts as desired and "NO" otherwise. Note that the output format is case-sensitive.

Sample output

```
Case #1: YES
Case #2: NO
Case #3: NO
```

Constraints

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

- $1 \leq T \leq 25$
- $1 \leq N \leq 1\,000$
- $1 \leq w_i, d_i \leq 1\,000\,000$

Time limit

5 seconds

7. Inverse Fibonacci

Introduction

The Fibonacci sequence is a famous sequence that appears frequently in nature. It is defined as follows:

$$F_0 = 1 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 2.$$

The first few terms are 1, 1, 2, 3, 5, 8, 13, 21.

One can create *generalised* Fibonacci sequences by replacing the first two terms. For example, if $F_0 = 2, F_1 = 1$ then the first few terms are 2, 1, 3, 4, 7, 11, 18. For the purposes of this problem, the first two terms must be integers strictly greater than zero.

Task

Archaeologists have discovered some numbers inscribed on the walls of an ancient site. They suspect that generalised Fibonacci sequences were important to this civilisation. For each inscribed number, they want you to find a generalised Fibonacci sequence that contains it.

A given number will be found in many generalised Fibonacci sequences. You must find the sequence containing the number for which:

- the number occurs as late in the sequence as possible i.e., if the number is F_n, then n must be as large as possible;
- if there is a tie, the sequence for which F_0 is as small as possible.

Example

Suppose one of the numbers found is 100. It is F_7 in the sequence $6, 4, 10, 14, 24, 38, 62, 100$, and there is no better sequence according to the tie-breaking rules above.

Input

The first line of input contains T, the number of inscribed numbers. The following T lines each contain one of these integer numbers.

Sample input

```
3
13
100
1234567
```

Output

For each inscribed number, output a line of the form

```
Case #X: A B n
```

where X is the test case number, starting from 1; A and B are the first two terms of the generalised Fibonacci sequence (F_0 and F_1); and F_n is the inscribed number.

Sample output

```
Case #1: 1 1 6
Case #2: 6 4 7
Case #3: 922 681 16
```

Constraints

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

- $1 \leq T \leq 50$
- $2 \leq$ each inscribed number $\leq 1\,000\,000\,000$

Time limit

2 seconds