Problem#01

Given two strings `s` and `t`, return `true` *if* `t` *is an anagram of* `s`*, and* `false` otherwise.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:

```
Input: s = "anagram", t = "nagaram"

Output: true
```

Example 2:

```
Input: s = "rat", t = "car"

Output: false
```

Constraints:

- `1 <= s.length, t.length <= 5 * 10^4`
- `s` and `t` consist of lowercase English letters.

Problem#02

We consider a number to be beautiful if it consists only of the digit 1 repeated one or more times. Not all numbers are beautiful, but we can make any base 10 positive integer beautiful by writing it in another base.

Given an integer **N**, can you find a base **B** (with B > 1) to write it in such that all of its digits become **1**? If there are multiple bases that satisfy this property, choose the one that maximizes the number of **1** digit.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with an integer **N**.

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 base described in the problem statement.

Sample

Sample Input

```
2
3
13
```

Sample Output

```
Case #1: 2
Case #2: 3
```

In case #1, the optimal solution is to write 3 as 11 in base 2.

In case #2, the optimal solution is to write 13 as 111 in base 3. Note that we could also write 13 as 11 in base 12 or as 1 in base 13, but neither of those representations has as many 1s.

Problem#03

Professor Shekhu has another problem for Akki today. He has given him three positive integers **A**, **N** and **P** and wants him to calculate the remainder when $\mathbf{A}^{\mathbf{N!}}$ is divided by **P**. As usual, **N!** denotes the product of the first **N** positive integers.

Input

The first line of the input gives the number of test cases, T. **T** lines follow. Each line contains three integers **A**, **N** and **P**, as described above.

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 answer.

Sample

Sample Input

```
2
2 1 2
3 3 2
```

Sample Output

```
Case #1: 0
Case #2: 1
```

In Sample Case #1, the answer is the remainder when $2^{1!} = 2$ is divided by 2, which is 0.

In Sample Case #2, the answer is the remainder when $3^{3!} = 3^6 = 729$ is divided by 2, which is 1.

Problem#04

There are n cities and m flight connections between them. Your task is to add new flights so that it will be possible to travel from any city to any other city. What is the minimum number of new flights required?

Input

The first input line has two integers n and m: the number of cities and flights. The cities are numbered 1, 2,…,n.

After this, there are m lines describing the flights. Each line has two integers a and b: there is a flight from city a to city b. All flights are one-way flights.

Output

First print an integer kk: the required number of new flights. After this, print kk lines describing the new flights. You can print any valid solution.

Constraints

1 ≤ n ≤ 1051 ≤ n ≤105

1 ≤ m ≤2⋅1051 ≤ m ≤ 2⋅105

1 ≤ a,b ≤n1 ≤ a,b ≤ n

Example

Input:

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

Output:

```
1
4 2
```

Problem#05

Alice likes reading and buys a lot of books. She stores her books in two boxes; each box is labeled with a pattern that matches the titles of all of the books stored in that box. A pattern consists of only uppercase/lowercase English alphabet letters and stars (*). A star can match between zero and four letters. For example, books with the titles `GoneGirl` and `GoneTomorrow` can be put in a box with the pattern `Gone**`, but books with the titles `TheGoneGirl`, `Gonetomorrow`, and `GoneWithTheWind` cannot.

Alice is wondering whether there is any book that could be stored in either of the boxes. That is, she wonders if there is a title that matches both boxes' patterns.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each consists of two lines; each line has one string in which each character is either an uppercase/lowercase English letter or *.

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 `TRUE` if there is a string that matches both patterns, or `FALSE` if not.

Sample

Sample Input

```
3
****
It
Shakes*e
S*speare
Shakes*e
*peare
```

Sample Output

```
Case #1: TRUE
Case #2: TRUE
Case #3: FALSE
```

In sample case #1, the title `It` matches both patterns. Note that it is possible for a * to match zero characters.

In sample case #2, the title `Shakespeare` matches both patterns.

In sample case #3, there is no title that matches both patterns. `Shakespeare`, for example, does not work because the * at the start of the `*peare` pattern cannot match six letters.