GoogleGames 2011
Pittsburgh

Submissions

Centauri Prime

7pt	Not attempted **28/30 users** correct (93%)
8pt	Not attempted **13/27 users** correct (48%)

Music Collection

8pt	Not attempted **12/12 users** correct (100%)
12pt	Not attempted **12/12 users** correct (100%)

Extreme Escalator Pogo

5pt	Not attempted **2/3 users** correct (67%)
10pt	Not attempted **1/2 users** correct (50%)

Top Scores

PeterLiang	45
trollmonkey1	30
prettypinkponies	30
jahooma	30
sjhakorea	30
jboning	20
Peterliang	20
WillDevanny	15
tjwilson	15
DFlat2	15

Problem A. Centauri Prime

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 1 7 points	Solve A-small-1
Small input 2 8 points	Solve A-small-2

Problem

Back in the old days before the creation of the mighty Centauri Republic, the planet Centauri Prime was split into several independent kingdoms. The kingdom of Mollaristan was ruled by king Loatold, while the kingdom of Auritania was under the rule of queen Elana. In fact, it just so happened that every kingdom whose name ended in a consonant was ruled by a king, while every kingdom whose name ended in a vowel was ruled by a queen. Also because of an amazing coincidence, all kingdoms whose named ended in the letter 'y' were constantly in a state of turmoil and were not ruled by anyone. Can you write a program that will determine the current rulers of several countries, given the countries' names?

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow, each one containing the name of one country. Country names will consist of only lower case English letters, starting with a capital letter. There will be no other characters on any line, and no empty lines.

Output

For each test case, output one line containing "Case #**x**: **C** is ruled by **Y.**", where **x** is the case number (starting from 1), **C** is the country name, and **Y** is either "a king", "a queen" or "nobody".

Be careful with capitalization and the terminating period. Your output must be in exactly this format. See the examples below.

Limits

1 ≤ **T** ≤ 300.

Small dataset

Each country name will have between 3 and 20 letters.

Large dataset

Each country name will have at most 100 letters.

Sample

Input	Output
3 Mollaristan Auritania Zizily	Case #1: Mollaristan is ruled by a king. Case #2: Auritania is ruled by a queen. Case #3: Zizily is ruled by nobody.

Problem B. Music Collection

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 1 8 points	Solve B-small-1
Small input 2 12 points	Solve B-small-2

Problem

Audio Phil has a huge music collection, and he is very particular about the songs he listens to. Each song has a name that is a string of characters. His music player has a search feature that lets Phil type a substring into the search box, and the player then lists all songs whose names contain the substring. If there is exactly one song that matches the search, then Phil can hit the Enter key to play that song.

Phil hates using the mouse, and he doesn't like typing too much, so he insists on always typing the shortest possible substring that will match exactly the one song that he wants to play at this moment. Could you help him find his optimal search query?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each one starts with a line containing a single number **N**. The next **N** lines each contain one song name -- these are all of the songs in Phil's collection.

Each song name will consist of only letters, spaces and the hyphen character (-). All songs in Phil's collection will be unique and at most 100 characters in length. Song names are case insensitive, so "dZihan" is the same is "Dzihan". The search algorithm is also case insensitive.

Output

For each test case, output one line containing "Case #**x**:", where **x** is the case number (starting from 1). After that, print **N** lines, one for each song in Phil's collection, in the order that the songs were given in the input. For each song, print the shortest string of characters that will uniquely find that song. If there are several correct answers, print the lexicographically smallest one. Put double quotes around each string. If there is no correct answer, print ":(" without the double quotes.

Note that upper case letters come lexicographically before lower case letters, hyphen comes before all letters, and space comes before hyphen.

Limits

$1 \le \mathbf{T} \le 100$.

Small dataset

$1 \le \mathbf{N} \le 10$.

Large dataset

$1 \le \mathbf{N} \le 100$.

Sample

Input	Output
5 6 A Perfect Circle - Gravity Aimee Mann - You Do Aqualung - Cinderella Arcade Fire - Haiti Art of Noise - Pleure ATB - Marrakech 2 Hybrid - Altitude Kings of Convenience - The Build-up 3 aaaaaaaabb aaaaaaabbb abababababab 3 butter fly	Case #1: "V" " D" "Q" " F" "S" "B" Case #2: "A" "C" Case #3: "AAAAAAAA" "BBB" "BA" Case #4: :(:(

```
butterfly                                "RF"
1                                        Case #5:
Unknown Artist - Track One               ""
```

GoogleGames 2011
Pittsburgh

A. Centauri Prime

B. Music Collection

C. Extreme Escalator Pogo

Questions asked

Problem C. Extreme Escalator Pogo

Small input 5 points	Solve C-small
Large input 10 points	Solve C-large

Problem

Robbit Downey Hopper is the famous inventor of the dangerous sport of Extreme Escalator Pogo. This sport is very dangerous; please do not try this at home, even if you do own an escalator. In fact, do not try this anywhere!

Extreme Escalator Pogo requires two things -- a jumping device called a pogo stick, and a fast moving escalator with **N** total steps that are rising up at a constant speed. Some of the steps are red, and the others are blue. Robbit starts by jumping onto a blue step in the middle of the escalator, on his pogo stick. He then continues to jump vertically while the escalator steps are moving underneath him. He must land only on the blue steps; if he lands on a red step, he is out, and his challenger Leepie Froggison gets her chance to jump next. Whoever manages to make the highest jump wins this game.

After touching his first blue step, Robbit always jumps up to a height of 1 and then lands on the next step. (It better be blue, or Robbit is out.) During each subsequent jump (except the first one), he has one of 3 choices:

- Dampen his jump to decrease the height by 1,
- Jump to the same height as in his current jump, or
- Amplify the jump to increase the height by 1.

A jump of height **H** takes exactly enough time for **H** escalator steps to pass underneath Robbit. Jumps of height zero are not allowed, but there is no limit on the maximum height. Escalator steps are on a loop that cycles forever.

Given **N** and the colour of each escalator step, can you help Robbit choose the best possible starting point and sequence of jumps to maximize the highest point that he can reach?

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each one contains an integer **N**, followed by an integer **K**, followed by **K** integers in the range between 1 and **N** inclusive that list the blue colored steps. All other steps are red. After step **N**-1, the next step is **N**, then step 1, then step 2, *etc*.

Output

For each test case, output one line containing "Case #**x**: **H**", where **x** is the case number (starting from 1) and **H** is the maximum height that Robbit can reach. If there is no limit, print "infinity" instead of **H**.

Limits

$1 \le \mathbf{T} \le 100$;
$3 \le \mathbf{N}$;
$1 \le \mathbf{K} \le \mathbf{N}$.
All step numbers will be different and sorted in increasing order.

Small dataset

$\mathbf{N} \le 10$;

Large dataset

$\mathbf{N} \le 10^9$;
$\mathbf{K} \le 1000$.

Sample

Input	Output
3 4 2 2 3 10 4 1 5 6 8 3 2 1 2	Case #1: 2 Case #2: 5 Case #3: infinity

Explanation

In the first sample case, the best Robbit can do is start with step number 2, then jump to step number 3 (height 1), then jump to height 2 and land on the red step number 1. The maximum height reached this way is 2.

In the second sample case, the best strategy is to start with step number 5, then go to 6 (height 1), 8 (height 2), 1 (height 3), again to 5 (height 4) and finally to 10 (height 5), which is red.

In sample case #3, Robbit can start with step number 1 and continue jumping forever. He can reach any height he wishes (unless he gets hungry or goes into Earth orbit).
