A - Bouncy Billiard Balls

The standard dimensions of a pool table are 44 x 88 inches. Imagine a table and ball in which there is no friction and no "English" on the ball. Upon being hit by the que stick, the que ball continues rolling indefinitely without curving and bounces against the walls with the angle of reflection being the same as the angle of incidence. In many cases, the ball will eventually complete some grand cycle and continue from where it started and in the same direction. Can you reason out what those conditions are? Fortunately, you don't have to worry about that here since all provided data will guarantee cycle repetition.

Given the initial coordinates of the ball when it is hit and its trajectory, your task is to determine how many bounces the ball makes before it begins to repeat a cycle. A corner bounce counts as two bounces, since this results in the ball retracing its path in reverse, the same as would happen if it were actually two bounces against each side of the corner that are infinitesimally close together. The trajectory will be given as two integers: the rise (dy) and the base (dx), with at least one of dy and dx being non-zero.

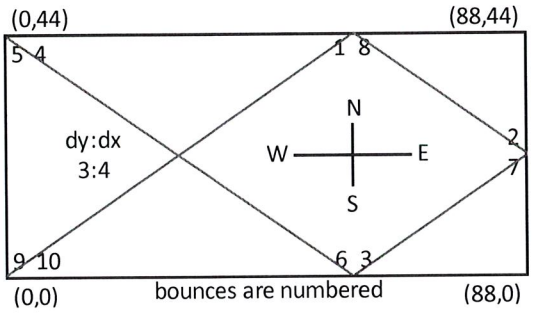
Input:
The first line contains a single integer indicating the number of cases. Each case consists of two lines of integers, the first containing the starting location in inches from the lower left corner: (x, y) 0 <= x <= 88, 0 <= y <= 44 and the second containing the trajectory as represented by (dx, dy). For your convenience, treat the diameter of the ball as being 0 so that it can be at position (0, 0), (44, 88) etc. without penetrating the wall. You may wish to think of the origin (0, 0) as being at the far southwest corner of the table.

Output:
For each case, print the case number and the number of bounces made by the ball before it repeats its cycle. Format as in the sample.

Sample Input:
```
2
44 22
0 1
0 0
4 3
```

Sample Output:
```
Case 1: 2
Case 2: 10
```

Case 2 is illustrated in the figure below. The path begins at the origin (0,0) with a trajectory of (4, 3), bounces 2/3 the way across the top, halfway down the right side, 2/3 the way across the bottom, and then at the upper-left corner (counted twice) at which it retraces its path back to the origin (also a corner so again count twice) after which it restarts its cycle. That's 5 + 5 = 10 bounces.



start = (44, 22)

176, 88

B - Double Up

In a mystical realm, your friend has been engrossed in a fascinating game in which the quest is to merge pairs of amulets in any one level to forge new amulets for the next echelon, with the ultimate goal of crafting the legendary level 10 amulet, called a Decadent, to claim victory. As the game commences, the player can employ a spell-like action called a duplex that instead of merging a pair of amulets, instead conjures a new level 1 and a new level 2 amulet to be placed onto the board. This can be repeated any number of times. Play begins with a board containing a random collection of amulets at each level.

The amulets at each echelon, starting with the first and lowest, have magical names. In order, they are Unident, Bident, Trident, Quadent, Pendent, Hexdent, Hepdent, Octadent, Nonadent, and the ultimate Decadent.

Engaged in this magical challenge, your friend seeks your insight to unravel the enigma: the minimum number of times this enchanting initial action (duplex) must be invoked in order to set in motion a chain of mergers, potentially involving al thel amulets, culminating in the creation of the coveted level 10 Decadent.

Embark on this arcane journey, where strategy intertwines with mystical lore, to assist your friend in achieving triumph in this captivating game.

Input: The first line will contain the number of rounds of the game to be played. Each round will consist of a single line containing 10 integers. The first integer represents how many level 1 Unidents exist at the beginning of the round, the second integer for level 2 Bidents, and so on up to and including level 10 Decadents. None of these integers will be larger than 9 or smaller than 0.

Output: For each round, print the minimum number of duplex actions that your friend needs to conjure up for the given round. Format as in the sample.

Sample Input:

1

3 1 4 2 1 0 1 1 1 0

1 0 1 0 1 1 1 1 0

Sample Output:

Case 1: 4

An illustration for the sample case is shown below. Four of the duplex actions are just sufficient to generate enough amulets that can be merged into the ultimate level 10 decadent. One Unident is left over. Mergers do not have to take place in any particular order, though are shown in the illustration working from the bottom-up.

After 4 duplexes the number of Unidents and Bidents increases from 3 and 1 to 7 and 5 respectively. 3 mergers of Unidents results in 6 of the Unidents merged into 3 additional Bidents, now 1 and 8 respectively. This process continues up the hierarchy until the 2 Nonadents merge to become a single Decadent, and the round is completed successfully. Starting with just 3 duplexes would not have been enough to climb all the way up.

Level Number	Amulet Type	Initial Count	After 4 Duplexes	3 Level 1 Mergers	4 Level 2 Mergers	4 Level 3 Mergers	3 Level 4 Mergers	2 Level 5 Mergers	1 Level 6 Merger	1 Level 7 Merger	1 Level 8 Merger	1 Level 1 Merger
10	Decadent	0	0									1
9	Nonadent	1	1								2	0
8	Octadent	1	1							2	0	0
7	Hepdent	1	1						2	0	0	0
6	Hexdent	0	0					2	0	0	0	0
5	Pendent	1	1				4	0	0	0	0	0
4	Quadent	2	2			6	0	0	0	0	0	0
3	Trident	4	4		8	0	0	0	0	0	0	0
2	Bident	1	5	8	0	0	0	0	0	0	0	0
1	Unident	3	7	1	1	1	1	1	1	1	1	1

D - Is it a Tree?

Squiggles, the aging squirrel, manages to find his way around, though his eyesight is dimming. When Squiggles comes up to what could be a tree, he checks it out as follows: He marks the base of the supposed trunk, then starts climbing. At various points, especially when Squiggles comes to a crotch (from which two or more branches may emerge), he marks the place and continues the climb. In case of a crotch with more than one branch, Squiggles selects an unexplored branch. This continues until he reaches the highest point (which he also marks) from which only leaves or nuts are found. Then he works his way back down to the previous crotch, and if there is another unexplored branch from there, he climbs up again. This up and down climbing continues until all crotches have been marked and all branches have been explored. If this is indeed a tree, Squiggles will end up back at the base from where he determines there are no more branches to explore. Note that the base of the tree could itself be a crotch from which multiple trunks split off.

For convenience we will indicate each marked position with a unique letter of the alphabet. You will be provided the sequence of letters used to label these positions in the order Squiggles first marks them and subsequently returns to them, so that Squiggles (and you) can verify that whatever he was climbing is or is not a tree. If it is indeed a tree, you are to also determine the greatest number of branches growing from any crotch plus the crotch itself.

One thing that may happen on Squiggles' grand tour that can indicate that the object is not a tree is that during an upward climb, Squiggles comes to a place that he had already marked. This would imply that the sequence of "branches" he is climbing forms a cycle.

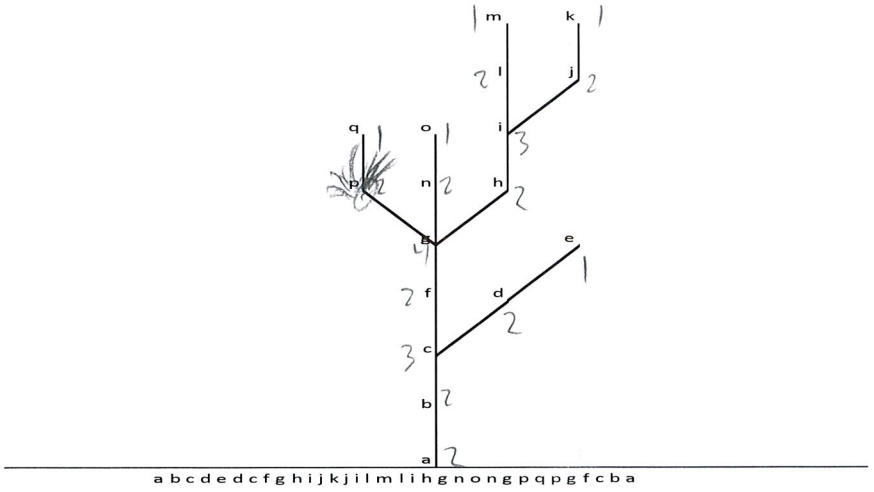
Input:
The number of cases will be indicated on the first line of input. Each case will consist of a string of lowercase letters representing the path followed on the supposed tree.

Output:
For each Case, print the degree if it is a tree or the message "Not a tree". Format as shown in the sample.

Sample Input:
2
abcdedcfghijkjilmlihgnongpqpgfcba
abca

Sample Output:
Case 1: Tree of degree 4
Case 2: Not a tree

Case 1 from the sample is illustrated in the following figure. Squiggles (recursively) climbs and backs down each of the branches, reaching e and backing down to c, and noticing that branch f is still unexplored, he climbs f and eventually reaches k before climbing back down to i. With l still unexplored, he climbs it and reaches m, and then backs down to g (no more unexplored branches coming from i). Finding unexplored branch n, he climbs it and reaches o before backing down again to g. With still another unexplored branch p, he climbs yet again and reaches q before returning to the base at a (there were no more unexplored branches coming from g or c so he keeps descending). He encountered g 4 times from different directions, the most times of any position (or crotch).



E - Kitten on the Keys

Piano keyboards are divided into "octaves" of seven relatively large white keys separated by smaller sets of two and three black keys that are set back a ways. See the figure on the next page. Each white key of an octave is assigned a letter of the alphabet from A to G, and the black keys can use the same letter plus a # (sharp) or b (flat). Striking each key produces a different note.

Kittens are known to jump onto piano keyboards and unintentionally play “tunes” as they scamper back and forth. In this problem we have a kitten pouncing on the light from a laser toy. The child holding the laser is trying to have the kitten play the notes that go with a song. From the figure, you may observe that it is easier for the kitten to hit the larger white notes, and besides, the laser light does not show up well on the smaller black keys. It is also easier to hit the B and C, and the E and F white keys because they are not separated by black keys. (Again, see the figure.)

A song can begin with any of the 12 black or white keys of any octave, and some tunes can extend beyond an octave. If you are given the notes of a song using a particular starting point (or key), you can play the same tune by shifting (or transposing) to the right (higher pitch) or to the left (lower pitch). Each successive key changes the note by a "half step".

For example, the start of “Twinkle, Twinkle Little Star” could be C, C, E, E, G, G, F, or transposing up by two half steps, D, D, F#, F#, A, A, G, noting that F# (read as F sharp) can also be read as Gb (or G flat).

You will be given the start of a song using an arbitrary key. Your task is to determine the starting key that yields the tune using the fewest black keys and white keys that are adjacent to two black keys. This is the same as asking for the most occurrences of E and F, and B and C.

Input:
Rather than have you deal with all the letter names and the flats and sharps, we'll simply number the keys. From the figure, let the middle C note with the yellow mark be note 0, then number the keys up and down as shown from -12 to 13. The "Twinkle, Twinkle" tune from above is now 0, 0, 4, 4, 7, 7, 5.

The first line of input will contain the number of songs to transpose. The first line for a song will contain the title of the tune, which you will send to the output, but otherwise can ignore. The second line for a song will contain the number of notes of the tune being presented. Each following line contains a single integer ranging from -39 through 48 (which can represent any note on a full size piano – the keys on the sample partial keyboard shown on the next page only go from -12 through 13).

Output:
For each song print 2 lines, formatted as shown in the sample: On the first line display the title of the tune, as provided in the input. On the second line, display the note number (from -39 through 48), which when used as the starting note for the tune, uses the minimum number black keys and white keys that are bordered on both sides by black keys. The transposition must fit entirely in the specified range of a piano. If there is more than one solution, choose the one starting closest to 0. If there is still a tie, choose the one with a positive value.

Sample Input:
1
Twinkle, Twinkle Little Star
7
1
1
5
5
8
8
6

Sample Output:
Case 1: Twinkle, Twinkle Little Star
0

The sample input with 7 notes starts the tune at note #1 which is a C# (see figure). This would result in 5 black keys (and no keys between black keys) for a total of 5. By transposing down by a half step to start at note #0 (which is C), the total drops to 2 (the white G key twice which is between 2 black keys.)

Easier keys to hit: