Objective:

- This assignment convers the issues related to rooted trees.

Challenge-1

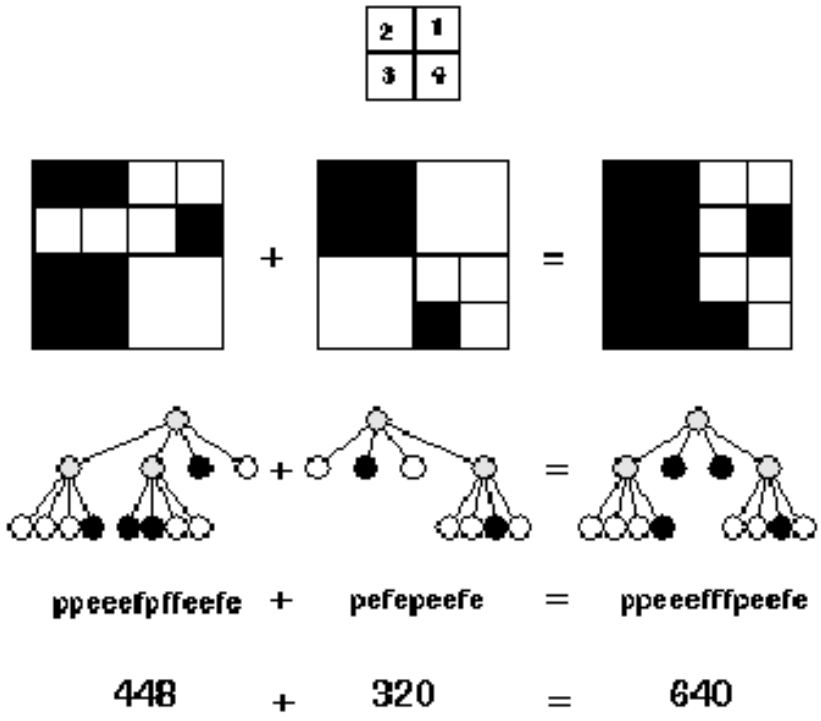
A quadtree is a representation format used to encode images. The fundamental idea behind the quadtree is that any image can be split into four quadrants. Each quadrant may again be split in four sub quadrants, etc. In the quadtree, the image is represented by a parent node, while the four quadrants are represented by four child nodes, in a predetermined order.

Of course, if the whole image is a single color, it can be represented by a quadtree consisting of a single node. In general, a quadrant needs only to be subdivided if it consists of pixels of different colors. As a result, the quadtree need not be of uniform depth.

A modern computer artist works with black-and-white images of 32*32 units, for a total of 1024 pixels per image. One of the operations he performs is adding two images together, to form a new image. In the resulting image a pixel is black if it was black in at least one of the component images, otherwise it is white.

This particular artist believes in what he calls the *preferred fullness*: for an image to be interesting (i.e. to sell for big bucks) the most important property is the number of filled (black) pixels in the image. So, before adding two images together, he would like to know how many pixels will be black in the resulting image. Your job is to write a program that, given the quadtree representation of two images, calculates the number of pixels that are black in the image, which is the result of adding the two images together.

In the figure, the first example is shown (from top to bottom) as image, quadtree, pre-order string (defined below) and number of pixels. The quadrant numbering is shown at the top of the figure.



Input Specification

The first line of input specifies the number of test cases (*N*) your program has to process.

The input for each test case is two strings, each string on its own line. The string is the pre-order representation of a quadtree, in which the letter 'p' indicates a parent node, the letter 'f' (full) a black quadrant and the letter 'e' (empty) a white quadrant. It is guaranteed that each string represents a valid quadtree, while the depth of the tree is not more than 5 (because each pixel has only one color).

Output Specification

For each test case, print on one line the text 'There are *X* black pixels.', where *X* is the number of black pixels in the resulting image.

Example Input

3
ppeeefpffeefe
pefepeefe
peeef
peefe
peeef
peepefefe

Example Output

There are 640 black pixels.
There are 512 black pixels.
There are 384 black pixels.

Challenge-2

You are building a web site for your clients to access, and want to create a welcome screen that serves as a starting point where they can view various pieces of data.

All of your data is stored hierarchically in folders, much like a file system. You are given a description of the folders. The description of a folder includes the id of the parent folder, a space, and a plus-sign-delimited list of users who have permission to view that folder. The root level folder, 0, is its own parent, and all folders are descendants of the root node. Additionally, you are given several usernames of the users who will be accessing the web site.

For each user, you are to determine which folder should serve as their "home folder". A user's home folder is defined as the deepest folder which contains all of the folders the user can access.

Suppose the data is very simple, such that folders looks like this:

folder 0: 0 Administrator
folder 1: 0 Joe+Phil
folder 2: 0 Joe
and your users are Administrator, Joe, and Phil. Clearly, the Administrator's home folder is 0, since he can access the root node. Similarly, Phil's home folder is 1, since he can only access folder 1. But Joe can access folders 1 and 2, so his home folder must be folder 0, which contains folders 1 and 2.

Input

The first line of the input contains the number F of folders (between 1 and 50 inclusive). Next F lines contain the descriptions of the folders. The folders are numbered from 0 to F-1 in the order in which they appear in the input. Each of these lines will be of the form "[parent] [user list]" (quotes added for clarity). Parent is an integer between 0 and F-1, leading zeroes are permitted. User list cannot be empty, but may contain repeats. The input defines a valid tree structure.

The next line of the input contains the number U of users to check (between 1 and 50 inclusive). Each username in the input contains only letters A-Z, a-z and is between 1 and 50 characters inclusive.

You may assume that no line in the input file exceeds 1000 characters in length.

Output

For each user in the list of usernames print a single line containing a single integer -- the number of the home folder for the corresponding user. Users who have access to no folders are assigned a home folder of -1.

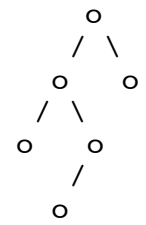
Example

Input	*output*
3 0 Admin 0 Joe+Phil 0 Joe 3 Admin Joe	0 0 1

Phil	
input 12 0 Admin 0 Jeff 1 Mark+Tim+Tim 1 Tim+Jeff 0 Dan 4 Ed 4 Tom 4 Kyle+Ed 0 Ben 8 Rich 8 Sam 8 Tim 4 Jeff Ed Tim Steve	***output*** 1 4 0 -1
1 0 Admin Peter Paul Mary	***output*** -1 -1 -1
3 0 Admin 0 Bob+Joe+Bob 0 Joe Joe Bob	***output*** 0 1

Challenge-3

Define the height of a binary tree to be the number of nodes in the longest path from the root to a leaf. The empty tree is considered to have height 0. A node is **k**-balanced if its left and right subtrees differ in height by at most **k**. A tree is **k**-balanced if all of its nodes are **k**-balanced. The empty tree is considered to be **k**-balanced.
For example, the tree below has height 4.



This tree is 2-balanced but not 1-balanced, because the left subtree of the root has height 3 and the right subtree of the root has height 1.
Your task is to write a method that takes a balance factor **k** and a number of nodes **n** and returns the maximum height of a **k**-balanced tree with **n** nodes.

Constraints

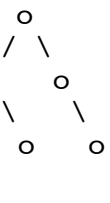
k is between 1 and 100, inclusive.
n is between 1 and 1000000, inclusive.

Examples

0)

1
7
returns: 4

A tree that achieves the maximum height for 7 nodes and balance factor 1 is



1)

2
40
returns: 9

2)

10
5
returns: 5

With **k**=10, a tree of size 5 can be completely linear (eg, every right subtree is empty) without violating the balance factor.

How to submit?

1. Remember it's a group assignment so hopefully you know the submission guidelines.

Learn from yesterday, live for today, hope for tomorrow. The important thing is to not stop questioning.
[Albert Einstein]