University of Manitoba
Open Programming Contest
September 27, 2008

General Instructions:

1. Submit solutions using the PC^2 software, as demonstrated.
2. The questions are **not** listed in order of difficulty.
3. All input should be read from standard input.
4. Remember that there is a 20-minute penalty for submitting an incorrect solution to the judges.

Problem 1 – Got perm?

Tony Earll has a series of integer sequences, and needs to know if each one is a permutation of the numbers from 1 to n (for some n) or not. If the input is such a permutation, he would like to know that, but otherwise, he needs the smallest integer which either does not appear in the list or which appears more than once.

Input

The first line gives n, the number of tests in the input. The next n lines consist of one or more positive numbers.

Output

For each line except the first, output the word PERM if the list of integers is a permutation of the first m integers for some m ≥ 1. Otherwise, output the smallest value that does not appear or appears more than once in the list.

Sample Input	Sample Output
`7`	`PERM`
`1 2 3`	`3`
`1 2 4`	`PERM`
`4 1 2 3`	`4`
`5 1 2 3`	`PERM`
`7 8 9 1 2 5 4 3 6`	`1`
`10 11 12 13 14 15 16 2 3 4 5 6 7 8 9`	`2`
`2 1 4 5 6 3 7 8 2`	

Problem 2 – Sequences of Palindromes

A palindrome is any word that is the same when spelled forwards and backwards. For instance, racecar is a palindrome, as are bob and noon.

In this question, you are asked to identify the longest (contiguous) substring of a string which is a sequence of palindromes (each of which has length two or more). By sequence of palindromes, we mean several (possibly different) palindromes each appearing next to each other in the string. For instance, the input

AABABCCBBCAA

has the substring AABABCCBB (AA BAB CC BB) as the longest sequence of palindromes.

Input

The input consists of an unknown number of strings, each on a separate line. Each string is a separate test case. You may assume each string is at most eighty characters long.

Output

For each case, print out the longest substring which is a sequence of palindromes (each of length at least two) on a separate line. If there is a tie between two sequences of palindromes of the same length, you should print out the substring which begins first in the string. If there is no palindromic substring of the input string, then print the single character ~ (the tilde) on a line to the output (You may assume that the input file contains no tildes).

Sample Input	Sample Output
AABABCCBBCAA AACBBB AACBBDEE AACBBCDD ABC	AABABCCBB BBB AA AACBBCDD ~

Problem 3 - Goldbach Conjecture

The Goldbach Conjecture was originally proposed by the Prussian mathematician Christian Goldbach in 1742 and is one of the oldest unsolved problems in mathematics today. The conjecture states that for every even number $m \geq 4$, there are prime numbers p and q such that

$$p + q = m$$

For example, for $m = 10$ we have 2 unique pairs of prime numbers that satisfies the condition in the conjecture:

$$3 + 7 = 10$$
$$5 + 5 = 10$$

Your task is to write a program that determines the number of unique prime number pairs that satisfies the conjecture for a given *m*.

Input

The first line of the input is an integer *N* followed by *N* test cases. Each test case consists of one integer, $2 \leq m < 100{,}000$.

Output

For each test case, print the number of unique prime number pairs that satisfies the conjecture.

Sample Input	Sample Output

```
4
2
10
20
6
```
	```
	0
	2
	2
	1
	```

Problem 4 – AIMP: ASCII Art Manipulation Program

```
  .                    /.
  \`.                 / |
   \ `.             .'  .
    \  `.         .'    .
     \   `- - - ''     /
      _..   .--.      .
     :    `:   _|     |
 .--.`._(@;`._(@;     .
: )  \               .
 \    :          |  /
   -.-          ; .'
     ``-= ....-'-"
```

Wilber is an ASCII art aficionado, but would really like an editor for ASCII art. Wilber probably doesn't need to remind you that ASCII art are two-dimensional pictures made from ASCII characters, such as the example above (image of Wilber copyright Vijay Kumar/bravegnu.org, used under GNU GPL).

To help address Wilber's lack of ASCII art manipulation programs, he has asked you to start by writing an application which can rotate ASCII pictures.

Input

The input is a list of an unknown number of pictures. Each picture begins with its dimension n (1 ≤ n ≤ 80). The image is square and the number of rows and columns in the picture is n. The next integer represents the rotation: for simplicity, it is either 45 or 90. Rotations are done in a counter-clockwise direction.

After the dimensions and rotation, there are n lines of text, each of length n. This is the ASCII art to be rotated. The last line of input is 0 0 representing the end of input. It should not be processed.

Output

For each image, print out a line containing "Image #x", where x is the number of the image, starting from 1. This is followed by the rotated image, and then one blank line. The characters themselves are printed as-is (not rotated). Each character in the input image produces exactly one character in the output image. All positions in the output image that are not mapped to a non-space character should be represented as a space.

Sample Input	Sample Output

Sample Input

```
6 45
abcdef
123456
67890-
qwerty
uiop[]
zxcvbn
7 90
-------
-  *  -
- * * -
-*   *-
-*****-
-*   *-
-------
7 45
-------
-  *  -
- * * -
-*   *-
-*****-
-*   *-
-------
5 90
*****
  *
  *
  *
*****
0 0
```

Sample Output

```
Image #1
     f
    e 6
   d 5 -
  c 4 0 y
 b 3 9 t ]
a 2 8 r [ n
 1 7 e p b
  6 w o v
   q i c
    u x
     z

Image #2
-------
-  ***-
- * * -
-*  * -
- * * -
-  ***-
-------


Image #3
      -
     - -
    -   -
   -     -
  - * * * -
 -       * -
-   *   * * -
 -     *   -
  - * *   -
   - *   -
    - * -
     - -
      -

Image #4
*   *
*   *
*****
*   *
*   *
```

Problem 5 – shuvvl.com

Shuvvl.com is your completely new idea for a website where users vote for stories that they feel are interesting (a process you call *shuvvling* the story). The more shuvvls a story has from different users, the more visible the story will be. The highest honour for a story on shuvvl.com is to be shuvvled so much that it is featured on the front page of the website.

The problem is that sometimes groups of people make concerted efforts to shuvvl stories to promote an agenda. Ideally, stories featured on the front page of shuvvl.com should have a broad appeal and have been shuvvled by many distinct users. Your job is to write a program to detect likely incidents of organized shuvvling and prevent those stories from reaching the front page.

A likely shuvvling incident is defined as a set of two or more stories for which

1. there is one story in the set (say, story number m), which identifies the organized users. All the organized shuvvlers, and no one else, shuvvled story number m.
2. all stories in the set were shuvvled by every user who shuvvled story number m.
3. additionally, the number of shuvvls for each story in the set is at most 1.5 times the number of shuvvls of story number m.

The shuvvlers of story m represent the shuvvlers who are organizing to promote a story, while the extra shuvvls on other stories represent people who are genuinely shuvvling stories that just happen to be the subject of an organized shuvvling.

Input

The first line has two integers, representing the number of tests for incidents we are conducting, and m, the total number of users of shuvvl.com (Users are represented by an integer between 1 and m unique to them). Each test consists of the following structure: The first line has one integer n, representing the number of stories to consider (n ≤ 50). Each of the next n lines represent the set of users who shuvvled a story. Your website does not allow a user to shuvvl a story more than once. The stories in each test case are implicitly numbered 1 to n starting from the first story. Tests are independent.

Output

For each test, begin with the output “Test #x:” where x is the test number, starting from 1. Then print a list of the stories (by number) that are not part of a likely shuvvling incident (and so are suitable for the front page). Print the numerical ID of each story on a separate line.

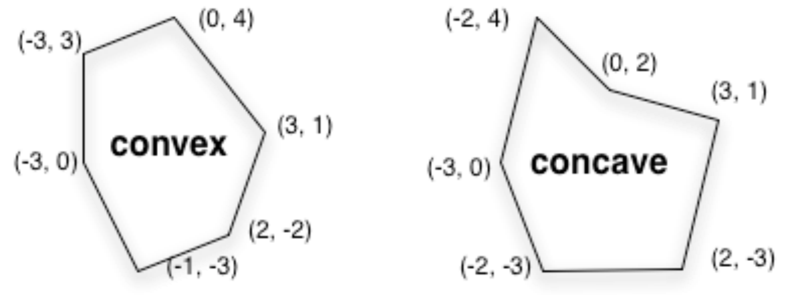
Sample Input	Sample Output

2 9
8
1 2 3
3 1 2 4
3 2 1 7 8 4 5 6
3 2 1 7
1 3 5 8
7 9
3 9 7
1 2 5 8
3
1 2 3
1 3 4
2 5 6
	Test #1:
	3
	5
	8
	Test #2:
	1
	2
	3

Problem 6 – Am I Convex or Not?

A convex polygon is one where the internal angles of the line segments at each corner is 180° or more. Convex polygons are less troublesome than concave polygons in computer graphics applications because they are easier to divide into triangles (the basic unit of output in computer graphics).

In two-dimensional space a polygon can be defined as a sequence of points specified with Cartesian coordinates, with an edge between each pair of points, and one more closing edge between the last and the first point in the sequence.



Input

The input file contains an unknown number of polygons. Each polygon includes an integer count of points *n* (*n* ≥ 4), followed by *n* lines containing floating-point *x* and *y* coordinates of a point. Edges exist between adjacent points, and between the last and first point. The "winding" of the polygon is not known, in that the points may be given in either clockwise or counterclockwise order.

Output

One line for each polygon: the word "true" for convex polygons, or "false" for concave polygons.

Sample Input	Sample Output

6
-1.0 -3.0
-3.0 0.0
-3.0 3.0
0.0 4.0
3.0 1.0
2.0 -2.0
6
-2.0 4.0
-3.0 0.0
-2.0 -3.0
2.0 -3.0
3.0 1.0
0.0 2.0
	true
	false