Problem A. Ancient Keyboard

Input file: `A.IN`
Program file: `A.cpp/A.c/A.dpr/A.java`

The scientists have found an ancient device that works in a strange way. The device has a keyboard and an output tape. The keyboard has 26 keys, with symbols '`A`' through '`Z`' on them. Each key has an LED on it (like the Caps Lock key on some keyboards). Each time you press a key, the LED on it toggles (changes its state from *off* to *on* or vice versa). All LEDs are *off* initially.

To study the output written on the tape, we consider the device in discrete time steps. Suppose we are in time t. If no LED is *on*, no output is written on the tape. If there are i LEDs *on*, the ith letter of the English alphabet is written on the tape. For example, if three LEDs are *on* at a time step, a letter '`C`' is written on the tape. This process repeats at every time step.

You are asked to write a program that simulates the ancient device.

Input

The input contains multiple test cases. The first line of the input, contains t, the number of test cases that follow. Each of the following t blocks, describes a test case.

The first line of each block contains one integer n ($0 \leq n \leq 26$). After this, there are n lines, each containing one capital alphabet letter, followed by two integers a and b, ($0 \leq a < b \leq 1000$). The capital letter shows the key pressed. The number a is the first time step at which the key is pressed and the number b is the second time step at which the key is pressed. During the interval $a, a+1, \ldots, b-1$, the LED of the key is *on*. You can assume that, in each test case, these letters are distinct.

Output

For each test case, output one line containing the output string that is written on the tape.

Sample input and output

A.IN	Standard Output
2 2 X 2 6 Y 4 9 3 A 1 5 B 4 8 C 9 10	AABBAAA AAABAAAA

B • THE DRUNK JAILER

Problem

A certain prison contains a long hall of ***n*** cells, each right next to each other. Each cell has a prisoner in it, and each cell is locked.

One night, the jailer gets bored and decides to play a game. For round 1 of the game, he takes a drink of whiskey, and then runs down the hall unlocking each cell. For round 2, he takes a drink of whiskey, and then runs down the hall locking every other cell (cells 2, 4, 6, …). For round 3, he takes a drink of whiskey, and then runs down the hall. He visits every third cell (cells 3, 6, 9, …). If the cell is locked, he unlocks it; if it is unlocked, he locks it. He repeats this for ***n*** rounds, takes a final drink, and passes out.

Some number of prisoners, possibly zero, realizes that their cells are unlocked and the jailer is incapacitated. They immediately escape.

Given the number of cells, determine how many prisoners escape jail.

Input

The first line of input contains a single positive integer. This is the number of lines that follow. Each of the following lines contains a single integer between **5** and **100**, inclusive, which is the number of cells ***n***.

Output

For each line, you must print out the number of prisoners that escape when the prison has ***n*** cells.

Example

Input	Output
2 5 100	2 10

Problem C: Symmetric Order

In your job at Albatross Circus Management (yes, it's run by a bunch of clowns), you have just finished writing a program whose output is a list of names in nondescending order by length (so that each name is at least as long as the one preceding it). However, your boss does not like the way the output looks, and instead wants the output to appear more symmetric, with the shorter strings at the top and bottom and the longer strings in the middle. His rule is that each pair of names belongs on opposite ends of the list, and the first name in the pair is always in the top part of the list. In the first example set below, Bo and Pat are the first pair, Jean and Kevin the second pair, etc.

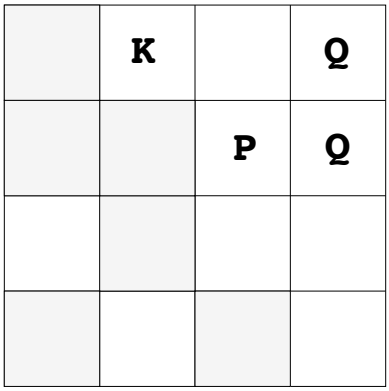
Input: The input consists of one or more sets of strings, followed by a final line containing only the value 0. Each set starts with a line containing an integer, *n*, which is the number of strings in the set, followed by *n* strings, one per line, sorted in nondescending order by length. None of the strings contain spaces. There is at least one and no more than 15 strings per set. Each string is at most 25 characters long.

Output: For each input set print "`SET` *n*" on a line, where *n* starts at 1, followed by the output set as shown in the sample output.

Example input:	Example output:
7 Bo Pat Jean Kevin Claude William Marybeth 6 Jim Ben Zoe Joey Frederick Annabelle 5 John Bill Fran Stan Cece 0	SET 1 Bo Jean Claude Marybeth William Kevin Pat SET 2 Jim Zoe Frederick Annabelle Joey Ben SET 3 John Fran Cece Stan Bill

Problem D: Queens, Knights and Pawns

You all are familiar with the famous 8-queens problem which asks you to place 8 queens on a chess board so no two attack each other. In this problem, you will be given locations of queens and knights and pawns and asked to find how many of the unoccupied squares on the board are not under attack from either a queen or a knight (or both). We'll call such squares "safe" squares. Here, pawns will only serve as blockers and have no capturing ability. The board below has 6 safe squares. (The shaded squares are safe.)



Recall that a knight moves to any unoccupied square that is on the opposite corner of a 2x3 rectangle from its current position; a queen moves to any square that is visible in any of the eight horizontal, vertical, and diagonal directions from the current position. Note that the movement of a queen can be blocked by another piece, while a knight's movement can not.

Input

There will be multiple test cases. Each test case will consist of 4 lines. The first line will contain two integers n and m, indicating the dimensions of the board, giving rows and columns, respectively. Neither integer will exceed 1000. The next three lines will each be of the form

$k\ r_1\ c_1\ r_2\ c_2 \cdots r_k\ c_k$

indicating the location of the queens, knights and pawns, respectively. The numbering of the rows and columns will start at one. There will be no more than 100 of any one piece. Values of $n = m = 0$ indicate end of input.

Output

Each test case should generate one line of the form

`Board` b `has` s `safe squares.`

where b is the number of the board (starting at one) and you supply the correct value for s.

Sample Input

```
4 4
2 1 4 2 4
1 1 2
1 2 3
2 3
1 1 2
1 1 1
0
1000 1000
1 3 3
0
0
0 0
```

Sample Output

```
Board 1 has 6 safe squares.
Board 2 has 0 safe squares.
Board 3 has 996998 safe squares.
```

Problem E: Guessing Game

Stan and Ollie are playing a guessing game. Stan thinks of a number between 1 and 10 and Ollie guesses what the number might be. After each guess, Stan indicates whether Ollie's guess is too high, too low, or right on.

After playing several rounds, Ollie has become suspicious that Stan cheats; that is, that he changes the number between Ollie's guesses. To prepare his case against Stan, Ollie has recorded a transcript of several games. You are to determine whether or not each transcript proves that Stan is cheating.

Standard input consists of several transcripts. Each transcript consists of a number of paired guesses and responses. A guess is a line containing single integer between 1 and 10, and a response is a line containing "too high", "too low", or "right on". Each game ends with "right on". A line containing 0 follows the last transcript.

For each game, output a line "Stan is dishonest" if Stan's responses are inconsistent with the final guess and response. Otherwise, print "Stan may be honest".

Sample Input

```
10
too high
3
too low
4
too high
2
right on
5
too low
7
too high
6
right on
0
```

Output for Sample Input

```
Stan is dishonest
Stan may be honest
```

G. Cormack

Problem F- City Slickers

In the movie City Slickers, Mitch, Ed and Phil had to migrate a herd of cattle between two ranches across the Montana Desert. In the open, they took no time at all, but to cross a mountain was perilous, and took 1 full day.

What's the fastest way for them to navigate the herd through the mountains?

Input Specification:

The first line of input will contain a single integer n. n test cases will follow.

Each test case will begin with 2 integers r, c, each at least 2, but not more than 40. r rows of text will follow, each of size c characters. This text will represent the map.

The map will contain exactly two '`r`'s, representing the ranches. A '`.`' is an empty area; anything else is a mountain.

Output Specification:

You will output the minimum number of days it will take the trio to complete their cattle run. Squares are adjacent only in the four compass directions (North, South, East and West), and the herd may never leave the map.

Sample Input:

```
2
2 3
rw.
wxr
5 5
...n.
.rmnx
xxvn.
xkzr.
.y...
```

Sample Output:

```
1
2
```

Problem G: To and Fro

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is "There's no place like home on a snowy night" and there are five columns, Mo would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character 'x' to pad the message out to make a rectangle, although he could have used any letter.

Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2 ... 20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Sample Input

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Sample Output

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

Problem H: Hard to Believe, but True!

The fight goes on, whether to store numbers starting with their most significant digit or their least significant digit. Sometimes this is also called the "Endian War". The battleground dates far back into the early days of computer science. Joe Stoy, in his (by the way excellent) book "Denotational Semantics", tells following story:

> "The decision which way round the digits run is, of course, mathematically trivial. Indeed, one early British computer had numbers running from right to left (because the spot on an oscilloscope tube runs from left to right, but in serial logic the least significant digits are dealt with first). Turing used to mystify audiences at public lectures when, quite by accident, he would slip into this mode even for decimal arithmetic, and write things like `73+42=16`. The next version of the machine was made more conventional simply by crossing the *x*-deflection wires: this, however, worried the engineers, whose waveforms were all backwards. That problem was in turn solved by providing a little window so that the engineers (who tended to be behind the computer anyway) could view the oscilloscope screen from the back.
> [C. Strachey - private communication.]"

You will play the role of the audience and judge on the truth value of Turing's equations.

Input Specification

The input contains several test cases. Each specifies on a single line a Turing equation. A Turing equation has the form "*a*+*b*=*c*", where *a*, *b*, *c* are numbers made up of the digits *0*,...,*9*. Each number will consist of at most *7* digits. This includes possible leading or trailing zeros. The equation `"0+0=0"` will finish the input and has to be processed, too. The equations will not contain any spaces.

Output Specification

For each test case generate a line containing the word `"True"` or the word `"False"`, if the equation is true or false, respectively, in Turing's interpretation, i.e. the numbers being read backwards.

Sample Input

```
73+42=16
5+8=13
10+20=30
0001000+000200=00030
1234+5=1239
1+0=0
7000+8000=51
0+0=0
```

Sample Output

```
True
False
True
True
False
False
True
True
```