
Highway Sign

Program Name: sign.java

Input File: <none>

Highway work crews often have portable, programmable signs that will inform drivers of lane closures and other maintenance activities.

Write a program that will print a text message.

Input

This program requires no input.

Output

Output the statement, "Left lane closed ahead."

Example Output To Screen

Left lane closed ahead.

Graph Icks

Program Name: graph.java

Input File: graph.in

Some people these days will do anything to get into shape. They'll even create bar graphs of their dietary and exercise performance.

Write a program that will analyze a bar graph of daily vegetable consumption and determine how many vegetables were eaten over the entire period.

Input

The first line of input will consist of a single integer, n , indicating the number of weeks of bar graphs to analyze. The remaining lines of input contain those bar graphs, each of which spans five lines (because the maximum number of vegetables consumed in a given day is 5) and 7 columns (because each graph represents a single week).

The bar graphs are vertical and will consist of empty space (represented by periods) and portions of the bars (represented by pound signs).

Bars can have zero height.

Output

For each bar graph in the input, output the message, “ x vegetables consumed in week y ” where y is 1 for the first graph, 2 for the second, etc. and x is the total number of vegetables consumed in that week.

Example Input File

2

```
.....
.#.....
.#...#..
.#####
#####.
.....
.....##
.....##
.....##
.....##
.....##
```

Example Output To Screen

```
14 vegetables consumed in week 1
8 vegetables consumed in week 2
```

Untris

Program Name: untris.java

Input File: untris.in

Untris is similar to a version of Tetris where all pieces are 1x1 blocks. A player drops blocks down chosen columns. Whenever a row of blocks is completed (i.e., has no empty spaces), that row disappears and all the blocks above it will fall by 1 unit.

Write a program that will simulate an Untris game.

Input

The first line of input will consist of a single integer, n , indicating the number of data sets. Each data set consists of:

1. A 9x9 Untris game board starting position. Empty spaces are indicated by periods ('.'). Blocks are represented by pound signs ('#').
2. A line containing a single integer, m , indicating the number of moves. ($1 \leq m \leq 20$)
3. A line containing m integers representing the moves (the columns where an Untris player is dropping his blocks).

Note: While a column is full of blocks, no move will drop an additional block in that column. However, if the column is reduced in height by one or more disappearing rows, the column is no longer full and may have more blocks dropped into it.

Output

For each data set in the input, first print the message, "Data Set #X" where X is 1 for the first data set, 2 for the second, etc. Then print the final state of the 9x9 game board after all moves have been completed.

Example Input File

```
2
.....
.....
.....
.....
.....
.....
.....
#####.###
#####.###
3
1 6 6
.....#...
.....#...
.....#...
.....#...
.....#...
.....#...
.....#...
..#####.##
.#####.##
5
1 7 7 1 6
```

Example Output To Screen

```
Data Set #1
.....
.....
.....
.....
.....
.....
.....
.....
#.....
Data Set #2
.....#...
.....#...
.....#...
.....#...
.....#...
.....#...
.....#...
#.#####
```

338 Phone Home

Program Name: phone.java

Input File: phone.in

A common cell phone keypad layout is as follows:



To enter a particular letter into the cell phone, one would press its corresponding number a number of times equal to its position in the sequence for that number. For example, to enter the letter 'y', one would press the '9' key three times. Numbers are always the last position in the sequence, so, for example, to enter the number '9', one would press the '9' key five times. To enter two or more consecutive letters/numbers that correspond to the same number, it is necessary to pause between entering each of them. For example, to enter the text "hi", one would press the '4' key two times, pause, then press the '4' key three times. Write a program to map key presses on a phone to alphanumeric text.

Input

The first line of input will contain a single integer, n , indicating the number of data sets to process. The remainder of the input consists of those n data sets.

Each data set will consist of a single line containing a series of key presses (0-9) and pauses. Pauses will be indicated with a 'P'. Note that the '*' and '#' will not be used in the input and that the string representing key presses and pauses will be less than 100 characters in length.

Output

For each data set in the input display its alphanumeric text result.

Example Input File

```
4
999
99999
44P444
184446337777P777774447777P77777P777444P4P448
```

Example Output To Screen

```
y
9
hi
1times7is7right
```

Scheming Plans

Program Name: scheme.java

Input File: scheme.in

Your love of literature and computer science has culminated in a desire to write a program that will, given a poem, output its rhyme scheme. A rhyme scheme is a pattern of rhyming lines, referred to by using letters to indicate which lines rhyme. We can identify the rhyme scheme by assigning letters to each rhyme, beginning with “a” and proceeding through the alphabet. For example, the poem:

There was a young fellow of Wheeling
Endowed with such delicate feeling
When he read on the door
Do not spit on the floor
He jumped up and spat on the ceiling

would have the rhyme scheme “aabba”, as lines 1,2, and 5 rhyme with each other, and lines 3 and 4 rhyme with each other.

Input

The first line of input will consist of a single integer, n , indicating the number of datasets in the remaining input. Each dataset will consist of:

1. A line containing a single integer, $1 \leq m \leq 10$, indicating the number of lines in the poem.
2. m lines of the poem. Poems will consist solely of words and spaces (no punctuation), where a word is defined as a contiguous unit of letters.

Note: for the purposes of this problem, consider two words to rhyme if the latter half (rounded up) of their letters are identical, regardless of case. Following these rules, consider the third example input below. The latter halves of the last words in each line are: ght, ight, ght, ight. So even though a normal person can tell that these words really do rhyme, the algorithm of comparing the latter halves of the words can only tell that lines 1 and 3 match, and that lines 2 and 4 match, resulting in the rhyme scheme: abab.

Output

For each dataset (poem) in the input, output a single line containing its rhyme scheme.

Example Input File

```
3
5
There was a young fellow of Wheeling
Endowed with such delicate feeling
When he read on the door
Do not spit on the floor
He jumped up and spat on the ceiling
4
Here is the church
And here is the steeple
Open the door
And see all the people
4
Star Light Star bright
The first star I see tonight
I wish I may and I wish I might
Have the wish I wish tonight
```

Example Output To Screen

```
aabca
abcd
abab
```

Circular Logic

Program Name: logic.java

Input File: logic.in

Imagine that you have a computer monitor that can only display five lines of text. Also imagine that your only running program is a counting program that begins at 1 and counts upward, one number at a time. It prints the numbers on your monitor, one per line, wrapping back to the top line after writing the bottom one.

The only interaction you have with this program is the ability to hit the space bar to have it wrap early (i.e., before it has written the last line). This is called a 'reset'.

Each column of number below indicates what your computer monitor would show at each step as your program counted from 1 to 21 without any resets:

```
1  1  1  1  1  6  6  6  6  6 11 11 11 11 11 16 16 16 16 16 21
   2  2  2  2  2  7  7  7  7  7 12 12 12 12 12 17 17 17 17 17
      3  3  3  3  3  8  8  8  8  8 13 13 13 13 13 18 18 18 18
         4  4  4  4  4  9  9  9  9  9 14 14 14 14 14 19 19 19
            5  5  5  5  5 10 10 10 10 10 15 15 15 15 15 20 20
```

However, let's say that you decided to perform a reset after the 7, 12, and 18 were printed.

```
1  1  1  1  1  6  6  8  8  8  8  8 13 13 13 13 13 18 19 19 19
   2  2  2  2  2  7  7  9  9  9  9  9 14 14 14 14 14 20 20
      3  3  3  3  3  3  3 10 10 10 10 10 15 15 15 15 15 21
         4  4  4  4  4  4  4 11 11 11 11 11 16 16 16 16 16
            5  5  5  5  5  5  5 12 12 12 12 12 17 17 17 17
```

As you can see, the first reset caused the 8 to start over in the first row instead of the third. The second reset had no real effect since the 13 was going to be in the first row anyway. And the third reset caused the 19 to be printed on the first row instead of the second. The final result: 19, 20, 21, 16, 17, is considerably different from that in the example without any resets.

Write a program that can determine, given the final output on the monitor, the minimum number of resets needed to generate the final pattern.

Input

The first line of input will contain a single integer n , indicating the number of data sets. The remainder of the input consists of those n data sets. Each data set will consist of five lines, containing the final display of the monitor, as illustrated in the above examples.

Notes:

- All inputs are valid (i.e., could be generated by performing the correct resets).
- None of the final displays in the input will contain numbers larger than 25.
- Resets will not occur until after the number 5 is displayed.
- At most 10 resets will occur in any one data set.

Output

For each data set in the input display a single line containing the minimum number of resets needed to produce the final display.

Example Input File

6
21
17
18
19
20
19
20
21
16
17
6
2
3
4
5
7
2
3
4
5
9
10
3
4
5
11
2
3
4
5

Example Output To Screen

0
2
0
1
2
5