

Computer Science Competition

2006 State Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.
2. All problems have a value of 60 points.
3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.
4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

II. Point Values and Names of Problems

Number	Name	Point Value
Problem 1	Hello Earl!	60
Problem 2	Homework	60
Problem 3	Vending Machine	60
Problem 4	Sudoku Junior	60
Problem 5	Sudoku Senior	60
Problem 6	Just Print It	60
Problem 7	Biathlon	60
Problem 8	Advanced Biathlon	60
Problem 9	Patterns	60
Problem 10	Robo Maze	60
Problem 11	Pager Printout	60
Problem 12	Generalized Fibonacci Haikus	60
Total		720

Hello Earl!

Program Name: hello.java **Input File:** hello.in

Write a program that will print a greeting for a specified person.

Input

The input file will contain a single name, which will consist of 1-20 alphabetical characters.

Output

Output the statement, "Hello <name>!" where <name> is taken from the input file.

Example Input File

Earl

Example Output To Screen

Hello Earl!

Homework

Program Name: homework.java**Input File:** homework.in

Write a program to solve your little brother's math homework.

Input

The first line of input will contain a single integer n indicating the number of math problems to solve. The remainder of the input consists of those n math problems.

Each math problem is of the form:

$A \text{ operator } B$

A and B will both be nonnegative integers in the range $0 \leq A, B \leq 1000$. Operators will be one of +, -, *.

Output

For each math problem in the input, print the solved problem to the output, where C is the solution:

$A \text{ operator } B = C$

Example Input File

```
3
3 + 37
2 * 121
0 - 81
```

Example Output To Screen

```
3 + 37 = 40
2 * 121 = 242
0 - 81 = -81
```

Vending Machine

Program Name: vending.java**Input File:** vending.in

Write a program to determine if you can buy your favorite junk food from a vending machine with these items and prices (in cents):

Gum	25
Chips	75
Coke	65
Candy	55
Pretzels	60
Donuts	95

Input

The first line of input will contain a single integer n indicating the number of people that want to buy items from the vending machine.

Each of the following n lines represents one of the people in line and will consist of:

1. A single nonnegative integer indicating the number of cents the person has to spend.
2. The name of the item the person wants to buy. This will match one of the items available.

Output

For each person in line, determine whether or not they can afford their item and output a corresponding message.

If the person can afford the item, display:

You have <X> cents left.

where <X> is the number of cents remaining to the person after the purchase. If the person cannot afford the item, display:

You need <Y> cents more.

where <Y> is the number of extra cents needed for the person to be able to make their purchase.

Example Input File

```
2
45 Pretzels
58 Gum
```

Example Output To Screen

```
You need 15 cents more.
You have 33 cents left.
```

Problem 4**Sudoku Junior****60 Points**

Program Name: sudoku1.java

Input File: sudoku1.in

1	4	2	3
2	3	1	4
3	2	4	1
4	1	3	2

1	4	2	3
2	3	1	4
3	2	4	1
4	1	3	2

1	4	2	3
2	3	1	4
3	2	4	1
4	1	3	2

1	4	2	3
2	3	1	4
3	2	4	1
4	1	3	2

Write a program to determine if a given sudoku solution is valid.

This sudoku game is played on a 4x4 grid. When finished, each cell on the board will contain a single integer: 1, 2, 3, or 4. The numbers are placed so that each row, column, and quadrant of the board contains each of the four numbers exactly once. If the board meets this criteria, it is a valid solution.

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 n solved, though possibly invalid, sudoku boards. Note that the input is always a 4x4 grid consisting solely of the integers 1-4.

Output

For each sudoku board in the input, print one of the following statements declaring whether or not the solution is valid:

Board #X is valid

Board #X is invalid

where X is 1 for the first board, 2 for the second, etc.

Example Input File

```
3
1423
2314
3241
4132
2134
3421
1214
4321
1234
2341
3412
4123
```

Example Output To Screen

```
Board #1 is valid
Board #2 is invalid
Board #3 is invalid
```

Problem 5**60 Points****Sudoku Senior****Program Name:** sudoku2.java**Input File:** sudoku2.in

1			3
			4
	2		
4			

Write a program that solves sudoku puzzles.

The rules are the same as in the previous sudoku problem, and only one valid solution exists to each puzzle.

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 n unsolved sudoku boards. Blank cells are indicated by a period ('.').

Output

For each sudoku board in the input, display the following:

1. A single line, "Data Set #X" where X is 1 for the first board, 2 for the second, etc.
2. The solved sudoku board.

Example Input File

```
2
1..3
...4
.2..
4...
1..3
2...
..3.
...1
```

Example Output To Screen

```
Data Set #1
1423
2314
3241
4132
Data Set #2
1423
2314
4132
3241
```

Problem 6		60 Points
------------------	--	------------------

Program Name: print.java **Input File:** <none>

Write a program that will output the following two statements and then terminate:

```
This problem has no input.  
It just has output.
```

Example Output To Screen

```
This problem has no input.  
It just has output.
```

Program Name: biath1.java**Input File: biath1.in**

Calculate the times of competitors in an Olympic biathlon and award the gold, silver, and bronze medals.

The biathlon is an Olympic event with two components, cross- country skiing and rifle shooting. In the women's individual competition, competitors race 15km. All competitors stop four times at the firing range and must fire once at each of five targets. For each target missed, one minute is added to their total time.

Given the wall- clock time required for competitors to complete the course and the hit/miss results for their 20 firing range targets, calculate the total time for each competitor and determine the medal recipients by comparing the total times.

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:

1. A line containing a single integer, m , indicating the number of competing athletes (always at least 3).
2. One line for each athlete, containing the following space- separated tokens:
 - Name – no more than 20 characters
 - Wall- clock Time – number of seconds between course start and stop times, given to the nearest hundredth of a second.
 - Targets Hit – number of targets that were hit.

Output

For each data set in the input display the following:

1. A single line, "Data Set #X" where X is 1 for the first data set, 2 for the second, etc.
2. One line for each athlete, in the same order as in the input, consisting of the following space- separated tokens:
 - Name
 - Total Time (Wall- clock Time + penalty time for missed targets) to the hundredth of a second
 - If the athlete receives a medal, display 'GOLD', 'SILVER', or 'BRONZE', as appropriate.

Note: There will be no ties.

Example Input File

```
2
3
Beth 2500.95 18
Laura 2550.73 19
Holly 2400.33 15
4
Beth 2500.95 18
Laura 2550.73 19
Holly 2400.33 15
Zippy 1000.80 0
```

Example Output To Screen

```
Data Set #1
Beth 2620.95 SILVER
Laura 2610.73 GOLD
Holly 2700.33 BRONZE
Data Set #2
Beth 2620.95 BRONZE
Laura 2610.73 SILVER
Holly 2700.33
Zippy 2200.80 GOLD
```

Program Name: biath2.java**Input File: biath2.in**

Calculate the times of competitors in an Olympic biathlon and award the gold, silver, and bronze medals.

This is similar to the preceding problem, with the exception that hit/miss results are given for only the first 19 of 20 firing range targets. A bullet heading is all that is given for the final target, which is 50 meters away with a diameter of 45 millimeters. You must use the heading to determine if the last target was hit or missed.

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:

1. A line containing a single integer, m , indicating the number of competing athletes (always at least 3).
2. One line for each athlete, containing the following space-separated tokens:
 - Name – no more than 20 characters
 - Wall-clock Time – number of seconds between course start and stop times, given to the nearest hundredth of a second.
 - Targets Hit – number of the first 19 targets that were hit.
 - Bullet Heading – the heading of the final bullet fired at the final target, in the format (i,y) , where i is the angle of inclination and y is the angle of yaw, both in degrees with up to 4 decimal places of precision. Inclination is positive for up, negative for down. Yaw is positive for a shot to the right, negative to the left. Assume that the bullets travel in a straight line and that target center would be hit if the heading were $(0,0)$.

Note: Bullets are considered as points (i.e., they have no radius) for the purposes of this problem, and no bullet will exactly hit the edge of a target.

Output

For each data set in the input display the following:

1. A single line, "Data Set #X" where X is 1 for the first data set, 2 for the second, etc.
2. One line for each athlete, in the same order as in the input, consisting of the following space-separated tokens:
 - Name
 - Total Time (Wall-clock Time + penalty time for missed targets) to the hundredth of a second
 - If the athlete receives a medal, display 'GOLD', 'SILVER', or 'BRONZE', as appropriate.

Note: The sample data for this problem only matches that of the previous problem for purposes of comparison. The judge data will not follow this pattern.

Note: There will be no ties.

Example Input File

```
2
3
Beth 2500.95 17 (0,0)
Laura 2550.73 18 (0,0.02)
Holly 2400.33 15 (-0.1,0)
4
Beth 2500.95 17 (0,0)
Laura 2550.73 18 (0,0.02)
Holly 2400.33 15 (-0.1,0)
Zippy 1000.80 0 (1,-1)
```

Example Output To Screen

```
Data Set #1
Beth 2620.95 SILVER
Laura 2610.73 GOLD
Holly 2700.33 BRONZE
Data Set #2
Beth 2620.95 BRONZE
Laura 2610.73 SILVER
Holly 2700.33
Zippy 2200.80 GOLD
```

Program Name: pattern.java**Input File:** pattern.in

Find patterns in text strings.

Given a list of patterns and a list of text strings, find all the letters in each string that match any of the given patterns.

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:

1. A pair of integers p s indicating the number of patterns and sentences in this data set, respectively. Each of these values will be at least 1 and at most 10.
2. A set of p patterns, each on its own line. Each pattern is a non- empty series of up to 20 alphabetic and question mark characters. Note that patterns will not contain spaces.
3. A set of s sentences, each on its own line. Each sentence is a non- empty series of up to 80 alphabetic and space characters. Sentences will not have any form of punctuation.

For a pattern to match a given portion of a sentence, all characters in the pattern must match a corresponding sequence of characters in a sentence, with question marks being a valid match for any single alphabetical character.

Output

For each data set in the input display the following:

1. A single line, "Data Set #X" where X is 1 for the first data set, 2 for the second, etc.
2. One line for each sentence, in the same order as in the input. Output the same sentence, replacing those alphabetic characters that were NOT matched by any of the patterns with asterisks ('*'). Leave spaces unchanged.

Example Input File

```
2
4 2
pro?ram?????
?o?
?e
m
How many programmers does it take to change a light bulb
None because it is a hardware problem
6 3
laugh
??ob????
?oo
ma?e
??rd
??dg??
you should love hard problems
we make the judges solve them first
contestants should be laughing at their all too simple mistakes
```

Example Output To Screen

```
Data Set #1
How m*** *rog**mme** doe* ** **ke ** ****ge * *****
None be***se ** ** * *****re *roblem
Data Set #2
*** ***** **** hard problems
** make *** judges *****
***** ** laugh*** ** ***** ** too *****
```

Program Name: maze.java

Input File: maze.in

Move a robot through a maze using the given commands, and print a map showing the result.

The robot can only move forward and back. If it hits a wall (or the bounds of the maze) when moving, it simply stops executing the current command and goes on to the next.

The robot can also destroy wall sections using a powerful on-board laser. When activated, the laser will vaporize the 1x1 section of wall that is first in its direct line of sight (if any). When walls are vaporized, the position in the maze previously occupied by the wall should be considered the same as any other open space in the maze (i.e., the robot can now move through that space).

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:

1. One line containing a pair of integers, r and c , indicating the number of rows and columns in the maze where $1 \leq r, c \leq 10$.
2. r lines, each of length c , representing the maze and the robot's starting location and orientation. The maze consists only of walls ('#') and open space ('.'). In the maze is one robot, represented by one of the following characters that also indicates the robot's orientation: N- north, S- south, E- east, W- west.
3. One line containing a single integer, m , indicating the number of commands given to the robot ($20 > m > 0$).
4. m lines, each containing one of the following robot commands:

TURN RIGHT	Causes the robot to turn right (clockwise) 90 degrees.
TURN LEFT	Causes the robot to turn left (counter-clockwise) 90 degrees
FORWARD <X>	Causes the robot to attempt to move forward X spaces ($10 > X > 0$).
BACK <X>	Causes the robot to attempt to move back X spaces ($10 > X > 0$).
ZAP	Activates the robot's on-board laser to vaporize a wall.

Output

For each data set in the input display the following:

1. A single line, "Data Set #X" where X is 1 for the first data set, 2 for the second, etc.
2. A representation of the maze, formatted just as given in the input, which shows the maze as it appears after the robot processes all the given commands. Vaporized walls should appear as any other empty space in the maze.

Example Input File

```
2
5 5
#####
#...#
#.#.#
#.N.#
#####
10
FORWARD 9
TURN RIGHT
FORWARD 3
TURN LEFT
FORWARD 2
TURN LEFT
FORWARD 1
TURN LEFT
ZAP
FORWARD 1
7 9
###.....
.....##..
.....#..#
.E...##..
.....#..#
.....##..
###.....
9
FORWARD 4
TURN LEFT
ZAP
TURN RIGHT
ZAP
TURN RIGHT
ZAP
TURN RIGHT
FORWARD 3
```

Example Output To Screen

```
Data Set #1
#####
#...#
#.S.#
#...#
#####
Data Set #2
###.....
.....##..
.....#..
.W...#..
.....#..
.....##..
###.....
```

Pager Printout

Program Name: pager.java

Input File: pager.in

Text pagers need to deliver short messages and have a limited number of characters available on their display. They can deliver messages longer than the display width by scrolling the message from left to right.

Write a program that will show the display of the pager during each step of scrolling a given message on a display of the specified width.

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 that consists of a positive integer ≤ 20 followed by a quoted string. The integer specifies the width of a pager's display (in characters) and the string inside the quotes is the message to be displayed.

Output

For each data set in the input display the following:

1. A single line, "Data Set #X" where X is 1 for the first data set, 2 for the second, etc.
2. One line for the starting display value and one additional line showing the display for each character shift necessary until the last character of the message is displayed. Each display string should be surrounded by double quotes and padded at the right with spaces, if necessary, to fill the width of the display.

Example Input File

```
3
5 "Hello World!"
20 "I like steak."
1 "blue"
```

Example Output To Screen

```
Data Set #1
"Hello"
"ello "
"llo W"
"lo Wo"
"o Wor"
" Worl"
"World"
"orld!"
Data Set #2
"I like steak.      "
Data Set #3
"b"
"l"
"u"
"e"
```


Program Name: haiku.java

Input File: haiku.in

A generalized Fibonacci haiku is a poem with a syllable count by line that follows a generalized Fibonacci sequence. Given two integers, **a** and **b**, a generalized Fibonacci sequence can be defined mathematically as:

$$F_n := F(n) := \begin{cases} a & \text{if } n = 0; \\ b & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

For example, the poem

Gym bag.
Old worn socks.
Must wash them right now.

would fit the syllable count by line for a value of 2 for **a** and 3 for **b**, since it has a syllable count by line of 2/3/5 (the first three integers in the generalized Fibonacci sequence for these values). Given a poem, determine if the poem's syllable count by line matches a generalized Fibonacci sequence.

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:

1. A line containing a single integer, m , indicating the number of lines in the poem ($2 \leq m \leq 10$).
2. The next m lines will consist of the poem.

Output

For each data set in the input display the following:

1. If the poem's syllable count by line follows a generalized Fibonacci sequence, a single line in the format "a b" where a and b are the values as described in the formula above.
2. Otherwise, a single line "NOT A GENERALIZED FIBONACCI HAIKU".

Note: For the purposes of the problem, determine the number of syllables in a line by counting the number of occurrences of the letters a, e, i, o, u, and y without regard for case.

Example Input File

```
3
3
Gym bag.
Old worn socks.
Must wash them right now.
4
Just two.
Now we have nine syllables.
Not really, but hard to estimate.
This line is really long but we need to have twenty syllables.
7
One.
Two.
Oh, wait.
That is not right.
My math may be off now.
First line had two and the next had one.
Now I need eighteen but this line is too short.
```

Example Output To Screen

```
2 3
2 9
NOT A GENERALIZED FIBONACCI HAIKU
```