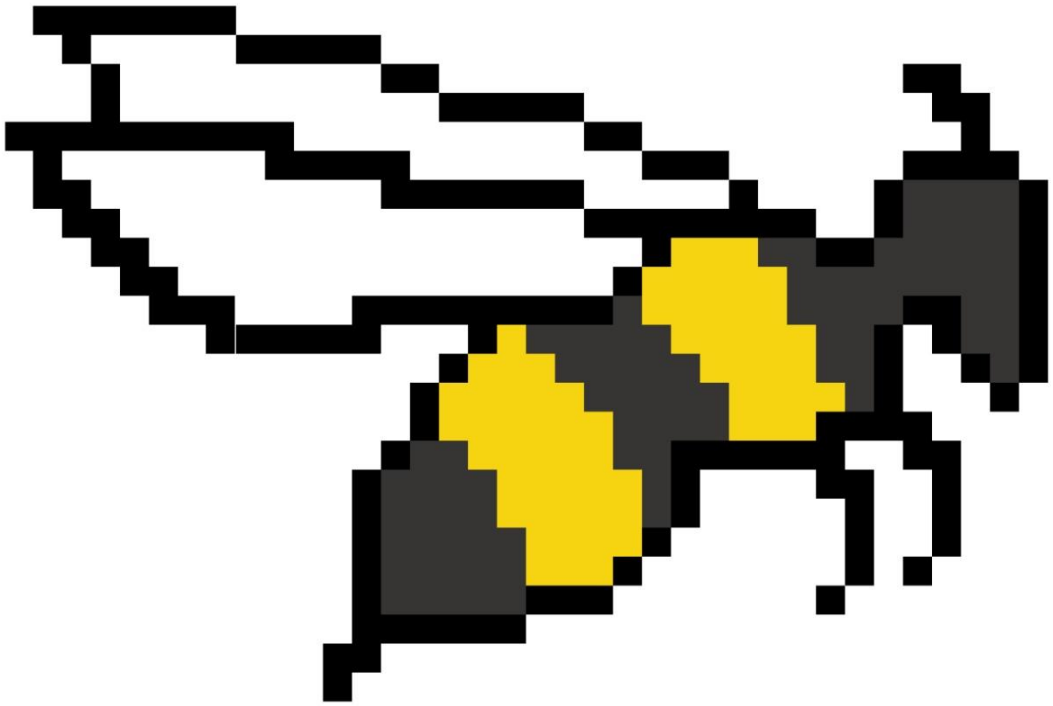


2010

BALDWIN-WALLACE PROGRAMMING CONTEST

DO NOT OPEN UNTIL INSTRUCTED!!!!!

Binary Arrays

Your father works as the manager of a paper manufacturer that has two sales teams of exactly 25 people each. At the end of each month, he determines which of the two sales teams sold the most paper for the month. He also determines the top three totals for each team and rewards a bonus if they sell more than the top three sales of the past month. Your dad's assistant collects the number of sales from each team on the last day of the month and gives the results to your dad. He then adds up the totals for each team, and the winners are announced the next day. Your dad's assistant had a mental break in the middle of the month, and began to think he was a robot, only speaking in "beeps" and "boops." Your dad went to human resources to ask what could be done, and he was told that his assistant could not be fired; the company (and more specifically, your father) would simply have to make changes to facilitate the assistant's "disability." Your father continued to give his assistant simple tasks to do, like printing and copying files, picking up coffee, and collecting the sales figures at the end of the month. The assistant dropped off the data at the end of the work day on your dad's desk while he was at a meeting. When your dad returned, he found a sheet of paper with two or three letter abbreviations of a worker's name, followed by a string of ones and zeros, which he assumes was the binary representations of the sales figures. With all of the members of the sales team gone for the day, and his assistant only making what he assumes are "robot noises," your dad brought home the sheet to see if you could help. Your dad was able to figure out that the abbreviations correspond to which team a person is on. That is, if a person's name is abbreviated with two letters, they are on Team A; if there are three letters, they are on Team B. He knows you've been taking classes in programming, and asks you to convert the binary numbers to base ten numbers. Knowing how badly he needs your help, you accept his request, but want to try and see what you can get out of the deal. You say you'll write a program that will generate a sales report for multiple months, displaying the sales numbers in base 10, reporting the winning team, and the top three sales figures on each team. In return, your dad must agree to let you borrow his Corvette for the weekend. Your dad, being desperate, accepts the offer, but says if you don't finish, you're grounded for a month.

Details of the input

The first line will contain a single integer, *n*, that indicates the number of months of sales data to be processed. The data for each month will consist of 50 lines of input. Each line will contain the two- or three-letter abbreviation of a worker's name. A colon then separates the name from the sales number in binary form for that person. For instance, if you are processing for 3 months, there will be 151 lines of input.

Details of the output

For each month, the first line of output should indicate which month is being reported in the form:

```
Month <number>:
```

The next two lines of output display the sales figure for Teams A and B, respectively, with individual sales

values separated by a space. The following line reports the total of Team A and the total of Team B separated by a space. The next line reports the relative sales of Team A and Team B in the following form:

```
Team A sold <more/less> than team B.
```

If the teams had the same sales, then the line should read:

```
The teams are equal.
```

Lastly, you are to list the top three sales figures for Team A and Team B, each organized from highest to lowest in the following fashion:

```
The top three sales for Team <letter> are: <value value value>
```

Sample Input (sales values are shown in two columns only to conserve space)

```
1
la: 11011                                   wer: 100111
hdo: 1111                                   jq: 100101
vc: 10010                                   lwc: 11101
fde: 10111                                  hbb: 111
abc: 101                                    ls: 10111
ki: 111                                     ns: 0
gj: 10001                                   der: 10101
red: 10101                                  al: 100101
br: 10011                                   odh: 101
hbo: 10111                                  cv: 11010
vf: 1010                                    edf: 101010
ere: 1111                                   cba: 11100
dac: 1110                                   ik: 11011
jl: 10101                                   jg: 11001
rt: 10011                                   pop: 10011
doo: 11011                                  ee: 101011
we: 10111                                   tot: 100101
opp: 10110                                  gp: 110
xo: 10010                                   dso: 10001
qpg: 1111                                   cpl: 11110
jut: 11011                                  le: 10111
cb: 10011                                   ok: 11001
sp: 1100                                    for: 11101
tin: 10101                                  ka: 101011
ca: 100011                                  hey: 10101
```

Sample Output

```
Month 1:
27 18 7 17 19 10 21 19 23 18 19 12 35 37 23 0 37 26 27 25 43 6 23 25 43
15 23 5 21 23 15 14 27 22 15 27 21 39 29 7 21 5 42 28 19 37 17 30 29 21
560 552
Team A sold more than Team B
The top three values for Team A are: 43 43 37
The top three values for Team B are: 42 39 37
```

Boxes of Boxes

Sammy Simpleton has a simple job at the Simply Stacked Box Company. Sammy must repeatedly fill large boxes with other boxes, making it simpler to store all of the boxes. For each large box, Sammy is told the dimensions of the box as well of the dimensions of three other boxes (call them boxes A, B, and C) from which Sammy chooses exactly one to use to pack the large box and he must orient the boxes all in the same direction. In order to simplify the stacking process, each large box should be packed with as many boxes as possible so that the least amount of space is left over. Occasionally, an error occurs and none of the three boxes can be used because they simply won't fit into the large box. The problem is, Sammy can't remember the simple formula he needs to figure out if any of the boxes will work and how much empty space would be left for each of the choices he could make. So, Sammy contacted you on his simple-to-use cell phone and asks you to write a program for him that can figure out which of the three boxes he should choose (if any).

Details of the input

For each large box that is to be filled, there will be four lines of input. The first line will be three integer values that represent the length, width, and height of the large box to be filled. The next three lines will also contain three integer values, representing the dimensions of boxes A, B, and C, respectively. Dimensions of all zero (0) for the large box will indicate the end of the input.

Details of the output

For each large box, there should be one line of output. When a solution can be found, it should be reported as:

```
A volume of <value> remains if box <letter> is used.
```

In the event of a tie, the alphabetically first box should be reported. For example, if both box A and C have the same remaining volume and it is less than box B, box A should be chosen.

If none of the boxes can be used, the output should read:

```
None of the boxes can be used.
```

Sample Input

```
6 7 8
2 3 4
3 3 3
6 2 2
6 7 8
6 7 9
9 2 3
3 10 1
0 0 0
```

Sample Output

```
A volume of 48 remains if box A is used.
None of the boxes can be used.
```

Fraction Bingo

Your younger brother is in fifth grade and his friends are having a fraction bingo competition in his math class. His favorite teacher has cleverly put him in charge of the competition and he accepted the position even though he does not know how to reduce fractions. So he has asked you to write a computer program to help him determine when a bingo will occur so he will appear smart. Your brother has explained how the bingo game will be played. Each of his classmates will receive a 5 by 5 bingo card with randomly assigned fractions already in lowest terms in each space. There is no free space on the cards. The bingo caller will be announcing a fraction (which may or may not be in lowest terms), and each student is then responsible for determining if they have an equivalent form of the fraction on their card. A player will only have a bingo if they have 5 of the called fractions in a row either horizontally, vertically, or diagonally. With your brother being the checker of the cards, he needs to know if one of the students has a bingo.

Details of the input

The first line of input will contain a positive integer, *n*, indicating the total number of rounds played. For each round, the first five lines of input will each contain five fractions of the form *p/q*, with each row representing the corresponding rows of the player's card. This next line has a single positive integer, *k*, which is the number of calls made during of the round. The final *k* lines for the round contain the fractions, one per line, that were called during the round. There may be more calls made than needed for a bingo to occur, in which case the remaining called fractions can be ignored.

Details of the output

There should be *n* lines of output, one for each round. If given round's card has a bingo the output should display the single word "BINGO"; otherwise, it should display the two words "NOT BINGO".

Sample input

```
1
1/1 1/2 1/3 1/4 1/5
1/6 1/7 1/8 1/9 1/10
3/4 3/5 3/7 3/11 3/13
5/7 5/9 5/17 5/19 5/21
7/9 7/11 7/13 7/15 7/31
7
6/8
10/14
2/10
28/36
1/1
2/20
3/18
```

Sample output

```
BINGO
```

Hidden Numbers

Your next door neighbor is a crazy conspiracy theorist. On morning he approaches you on your way to work with a problem that has been bothering him lately. He is convinced that there is a spy working within the local FBI precinct and that spy is leaking information to a foreign government! He says the messages have been cleverly encoded into a special system of numbers. Your neighbor explains to you that the spy is transmitting these numbers by placing their alphabetical representations into sentences within your local newspaper, as the spy has an "inside man" at the newspaper. He then goes on to say that the spy is going a step further and trying to throw off anyone who might detect him by only counting the largest number within the sentence to be used in his secret transmission. Determined to stop this spy before he brings down the world's top governments, your neighbor hands you a large list of sentences asking you to write a program that will automatically check them for "hidden numbers".

Details of the input

The first line will contain a positive integer, *n*, indicating the number of sentences to be analyzed. The next *n* lines will each contain a single sentence that needs to be searched for the highest value number in alphabetical format. Each sentence will contain no more than 80 characters including special characters and spaces. Your neighbor is only concerned with the numbers "ZERO" through "NINETYNINE" and anything higher than that is not important to him. In an attempt to throw you off further the newspaper man may include special characters within the text of the numbers (e.g., o#n$!e).

Details of the output

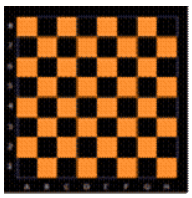
The program should output the highest number found within the sentence in all uppercase letters. If no numbers are found within the sentence your neighbor would like you to display "NONE FOUND" as output.

Sample input

```
1
S@e-v/en t     ee/ns like to drive ten fast cars!
```

Sample output

```
SEVENTEEN
```



Color Jumping

It is another wonderful day at the mall with your sister. The only problem is that your sister refuses to step on the same color tile twice in a row while she is walking. So your task is to find out if you can get to the store that you want to from your current position in the mall without leaving your sister behind (that is, by also alternating tile colors with each step). Also your sister refuses to walk in any other way except left, right, forward, and backwards – no diagonal moves. She is just weird like that.

Details of the input

The first line of input will be a single integer, i ($0<i<100$), which is the number of instances of the problem you will solve. The first line of each instance consists of two integers, j,k ($2<=j<=10$, $2<=k<=10$), separated by one space, which defines the size of the board for that instance. On the next j lines is the board for that case. Each line consists of k integers, each of which can be a 3, 2, 1, or 0 and represents a single tile. 3 represents the starting location, 2 is the finishing point, 1 represents a black tile, and 0 represents a white tile.

Details of the output

Your output should consist of one line per instance of the problem. This line should be of the form:

```
Case <number>: Path <found/not found>
```

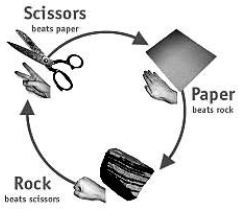
where *number* indicates the problem instance, starting with 1 for the first instance.

Sample input

```
2
5 5
3 0 1 0 1
1 1 1 1 0
1 1 1 1 1
1 0 0 0 0
1 1 1 1 2
5 5
3 0 1 1 1
1 1 1 1 0
1 1 1 1 1
1 0 0 0 0
1 1 1 1 2
```

Sample output

```
Case 1: Path Found
Case 2: Path Not Found
```



N-Way Rock Paper Scissors

Rock Paper Scissors is typically a two-player game where each player picks one of the three objects (rock, paper, or scissors), and the players both reveal their choice at the same time, usually by counting to three and then revealing. The act of showing your choice is sometimes called a "throw." If both players throw the same object, it is a tie and they play again until they throw different objects. The winner is then determined by these rules:

1. Rock beats scissors: The rock can smash the scissors.
2. Scissors beat paper: The scissors can cut the paper.
3. Paper beats rock: We have no idea why. Really.

More than two people can play Rock Paper Scissors at the same time, but things get slightly more complicated. Each player picks an object and they all throw at the same time (just like in two-player). Players may be eliminated on each throw, and the *remaining players* continue this process until only one remains. The rules for determining which players are eliminated on each throw depend on how many of the three objects are represented on that throw:

1. If only one of the three objects is represented then nobody is eliminated, just like a tie in the two-player version.
2. If exactly two of the three objects are represented, then one of those objects is stronger than the other object (according to the 3 rules above). All players throwing the *weaker* option are eliminated, regardless of which object had more people throwing it. For example, if there are 4 throwers and their throws are {rock, rock, rock, paper}, then the players throwing rock are eliminated.
3. If all three of the choices are represented and each has the same number of people throwing them, then nobody is eliminated (this is considered tie). If all three choices are represented *unevenly*, the group or groups with the most throwers move forward. For example, if there are 5 throwers and their throws are {rock, rock, rock, paper, scissors}, then the players throwing paper and scissors are eliminated. If there are 5 throwers and their throws are {rock, rock, paper, paper, scissors}, then only the player throwing scissors is eliminated.

Play continues amongst the players who have not been eliminated until only one player remains. Your job is to determine the winner of several *n*-way Rock Paper Scissors games.

Details of the Input

Each test case will begin with a line containing positive integers *n* and *m*, with *n* being at most 10 and *m* at most 15. The value in *n* represents the number of players. We guarantee that a winner will be determined in at most *m* throws (but perhaps less). The following *n* lines contain the following information for each player:

<name> <$throw_1$> <$throw_2$> … <$throw_m$>

The players name is a string of at most 20 characters, and each of the $throw_i$ values is a string from the set {"rock", "paper", "scissors"}. The list of *m* $throw_i$ values represent the order of objects that will be thrown by that player (even though they may be eliminated before the m^{th} throw).

Input values of $n = m = 0$ indicate the end of input and should not be processed.

Details of the Output

For each test case, output one line:

Round <case number> goes to <winner>!

where case number reflects which input case this output corresponds to, and winner is the name of the winner (the one player who is never eliminated according to the rules).

Sample Input

```
4 4
Lisa rock scissors paper rock
Johnny scissors scissors scissors scissors
Dave rock scissors paper paper
Tim rock scissors scissors paper
2 3
Dave rock paper scissors
Tim rock rock rock
0 0
```

Sample Output

```
Round 1 goes to Tim!
Round 2 goes to Dave!
```