University Interscholastic League

Computer Science Competition

2015 District 2 Programming Problem Set

DO NOT OPEN THIS PACKET UNTIL INSTRUCTED TO BEGIN!

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. Incorrect submissions receive a deduction of 5 points, but may be reworked and resubmitted. Deductions are only included in the team score for problems that are ultimately solved correctly.

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. Table of Contents

1. Abacus

Program Name: Abacus.java   Input File: none

Long before the advent of sophisticated graphics packages, there were artists who used just the symbols available on the keyboard to make their drawings. Some of their drawings were quite realistic and detailed. They always used mono-spaced fonts like Courier to draw their symbols. Since these fonts were fixed in width, it was easy to align them. Moreover, they used the Space key instead of the Tab key to denote spaces and the vertical bar (|) instead of the capital I to draw vertical lines. In this program you will have a chance to create an ASCII picture.

Input
There is no input for this problem.

Output
You are to output the abacus as shown below. Each character is of equal size (including spaces). Your output may look wrong and still be correct (it requires a fixed-width font to look the same as below), so make sure you are careful to follow the design exactly. You may assume that there are no trailing spaces on any given line. The picture begins with a blank line on the top and there is a blank line at the bottom.

Output to screen

```
_________________________________________
_________________________________________
| |(_) (_) (_) (_) (_) (_) (_) (_) (_)| |
| |_|___|___|___|___|___|___|___|___|_| |
| | |   |   |   |   |   |   |   |   | | |
| |(_) (_) (_) (_) (_) (_) (_) (_) (_)| |
| |(_) (_) (_) (_) (_) (_) (_) (_) (_)| |
| |(_) (_) (_) (_) (_) (_) (_) (_) (_)| |
| |(_)_(_)_(_)_(_)_(_)_(_)_(_)_(_)_(_)| |
_________________________________________
```

2. Burglary

Program Name: Burglary.java Input File: burglary.dat

You are organizing a heist of a large bank, and have obtained a blueprint of the bank's vault. The vault consists of a series of N rooms connected in a row. You start in the left-most room. Each room contains a certain amount of money and a vault door connecting it to the next room. The blueprints detail the amount of time it would take you to crack each vault door. You realize that if you take just the money from the first room and leave immediately, the police will not be able to catch you, and you will get to keep all of the money you took, But, if you try to open any vault doors, there is a 5% higher chance per door that the police will show up and you would have to dump the money and run, leaving you with nothing. For example, if you take the time to open 3 vault doors, there is a 15% chance the police will show up. Naturally, being a risky person, you do not want to take the safe route of only taking the money from the first room. You want to maximize the expected value of your heist. The expected value of a given heist plan is the probability you keep the money multiplied by the amount of money you stole. If the probability of the police showing up is p, then the probability of the police not showing up is (1-p), and if you would steal M dollars if you got away, then the expected value of that heist plan is (1-p)*M.

How many doors should you open to maximize your expected value? And what is the maximum expected value of money you can steal?

Input

The first line of input contains T, the number of test cases that follow.

The first line of each test case is the number of rooms N. The next line contains N space separated money values, the amount of money in each room.

Output

For each test case, output the number of doors to open to maximize your expected value, and the expected value of opening that many doors, separated by a space. The expected value should be rounded to the nearest cent.

Constraints

```
1 <= T <= 8
1 <= N <= 10
```

Example Input File

```
3
3
5.00 5.00 5.00
2
100.00 1.00
5
44.90 4.95 60.02 26.65 90.87
```

Example Output to Screen

```
2 13.50
0 100.00
4 181.91
```

Explanation of Output

In the first test case, if you do not open any doors, and just take the money in the first room, the expected value is 1.00 * 5.00 = 5.00. If you open the first door, and then run, the expected value is 0.95 * 10.00 = 9.50. If you open the first door, and then the second door, the expected value is 0.90 * 15.00 = 13.50. The maximum of these is 2 doors with 13.50.

In the second test case, if you take the money and run, the expected value is 1.00 * 100.00 = 100.00. If you open the first door and then run, the expected value is 0.95 * 101.00 = 95.95. The maximum of these is 0 doors with 100.00.

3. Christmas

Program Name: Christmas.java   Input File: christmas.dat

The alien planet Blarg celebrates gift giving at Christmas time just like we do on Earth. This Christmas season, the Blargian children really want 3 new toys: Bligs, Snorgs, and Kords. Since Blargian Santa's factory does not have the technology to make these, he has decided to make all of the rest of the toys in his factory, and outsource the making of these three toys to external contractors.
There are several potential contractors that Blargian Santa is looking at hiring. They are paid by the hour, and advertise quotas for each of the three toys that they will make, per hour.

Blargian Santa is bad at math, so he has sent a message to earth for help. Which contractor should he hire?

Input

The first line of input contains T, the number of test cases that follow.

The first line of each test case will be a single integer C, the number of cities on Blarg. The next C lines describe the amount of each toy the Blargian children in that city desire for Christmas. Each line contains 3 space separated integers: cB, cS, and cK, the number of Bligs, Snorgs, and Kords respectively.
The next line of each test case contains a single integer N, the number of suitable contractors Blargian Santa has found.
The next N lines describe each contractor. Each line contains a string, the name of the contractor, and then four space separated integers: B, S, K, and P, the number of Bligs, Snorgs, and Kords they will make per hour, and the price they charge per hour. They do not work partial hours.

Output

For each test case, print the name of the contractor that fulfills the toy order for the least amount of money, and the total price for that contractor, separated by a space. It is guaranteed that at least one contractor will be able to fulfill the order, and that there will be a unique "best" contractor.

Constraints

```
1 <= T <= 8
1 <= C, N <= 10
0 <= cB, cS, cK <= 1000
1 <= B, S, K, P <= 1000
```

Example Input File

```
2
2
0 0 1
0 0 2
2
A 0 0 2 40
B 0 0 1 25
5
57 82 54
57 83 73
43 40 47
61 40 90
3 13 24
3
A 56 2 26 10
B 45 11 57 52
C 63 19 10 33
```

Example Output to Screen

```
B 75
C 957
```

Explanation of Output

In the first case it seems the contractors only make Kords. Thankfully, the children do not want anything else this season. The total toy order is 0 Bligs, 0 Snorgs, and 3 Kords. Contractor A can produce 2 Kords an hour, so to fufill the order, they would have to work 2 hours, bringing the total cost for A to 80. Contractor B produces 1 Kord an hour, meaning it will take them 3 hours, but at 25 an hour, the total cost is 75. Therefore, Contractor B is cheaper by 5, and Blargian Santa should hire B.

In the second case the totals are much higher and though C produces the fewest number of Kords per hour. However, when calculated, the larger numbers of Bligs and Snorgs produced offsets the low Kord value making C the best choice.

4. General Fibonacci

Program Name: GeneralFib.java Input File: generalfib.dat

The boys in the field are intercepting messages from the evil General Fibonacci and sending them back home for you to decrypt. After years of code breaking, Commander Wythoff has discovered that General Fibonacci uses a formula to encrypt his numbers. The formula has two "base numbers" called `x` and `y` such that `f(1) = x` and `f(2) = y`. His formula also has "coefficient numbers," `a` and `b` which gives the function its definition for the rest of the numbers. `f(n) = a * f(n - 1) + b * f(n - 2)`.

Given `x`, `y`, `a`, `b`, and `n`, find `f(n)` in order to help the war effort.

Input

The file begins with an integer `T` $(1 \leq T \leq 100,$ the number of test cases. After that, `T` test cases follow. Each test case has 5 numbers in a single line, `x`, `y`, `a`, `b`, and `n`. It is guaranteed that $f(n) < 10^9$.

Output

For each test case, print a single line with the value of `f(n)`.

Example Input File

```
3
0 1 1 1 5
2 1 1 1 7
0 1 1 2 6
```

Example Output to Screen

```
3
18
11
```

Example Output Explanation

The first sequence begins `0`, `1`, `1`, `2`, `3`, `5`, `8`, `13`, … In this sequence, the fifth element is 3.

5. History Grading

Program Name: History.java   Input File: history.dat

On a history exam students have to arrange events in chronological order. Students who order all the events correctly receive full credit. However, how do we give partial credit to students who get a few of the events out of order? Here is one scheme for partial ordering that you are asked to implement.

You will award 1 point for each event in the longest (not necessarily contiguous) sequence of events that are in the correct order relative to each other. For example, if four events are correctly ordered 1 2 3 4 then the order 1 3 2 4 will receive a score of 3 (since the event sequences 1 2 4 and 1 3 4 are both in the correct order relative to each other). The order 4 3 2 1 will receive a score 1 (you are being generous).

Input

The first line will have two integers *m* and *n* separated by one or more spaces. The first integer *m* is the number of questions that you will have to grade, where $2 \leq m \leq 20$. The second integer *n* is the number of chronological events to order. After the first line, there will be *m* lines of input. Each line will have *n* integers in the range $1 \leq n \leq 1000$. On a given line of input no integer will be repeated. Each integer will be separated from other integers by one or more spaces. The correct order is the monotonically increasing sequence of the *n* integers.

Output

There will be *m* lines of output. Each line will have the score for that question.

Sample Input

```
5 5
1 2 3 4 5
5 4 3 2 1
33 29 47 52 19
403 256 38 575 14
3002 108 4267 5422 8543
```

Sample Output

```
5
1
3
2
4
```

6. ImageFlip

Program Name: ImageFlip.java Input File: imageflip.dat

Modern art is sometimes hard to appreciate. In the 1960s, the Museum of Modern Art in New York City hung a Henri Matisse painting upside-down for over a month! In order to help modern artists out, you need to install two buttons on each painting. One button flips the painting horizontally, and the other button flips paintings vertically. That way, museum visitors can view the paintings in the orientation they like.

Input
The file begins with an integer `T (1 ≤ T ≤ 10)`. After that, `T` test cases follow. Each test case begins with two integers and a character, `N`, `M`, and `C (1 ≤ N, M ≤ 30)`. `N` is the height of the field, `M` is the width of the field, and `C` is either `H` or `V`, representing horizontal and vertical flips respectively. After that follows `N` lines, each with `M` characters, representing blocks of color in the paintings. Each block of color is represented by a capital letter (`A-Z`).

Output
For each test case, print the painting reflected appropriately. Print a blank line between test cases.

Example Input File
```
2
2 2 H
AB
CD
3 3 V
XYZ
WAB
CDE
```

Example Output to Screen
```
BA
DC

CDE
WAB
XYZ
```

7. Jousting

Program Name: Joust.java   Input File: joust.dat

You are organizing a jousting tournament for the knights in your kingdom.

The tournament is a single elimination tournament, so it consists of a number of rounds. There are N knights who have signed up for the tournament. In this tournament, only the top 3 knights get the prize money.
Say there are R knights left in each round. The R knights line up against the wall of the stadium. The knights are paired off into R/2 pairs, with the first two on the left facing each other, then the next two facing each other, etc. The pairs then joust, and the winners return to their same spots on the wall, and the losers exit the stadium. Approximately half of the remaining knights are defeated each round, up until the round with 4 knights, or the semifinals.
The two knights that win in the semifinals will advance to the finals. The winner of the finals gets first place, and the loser of the finals gets second place. Since there are only 3 prizes, the two knights that lost in the semifinals must play an extra match, with the winner claiming third place.
You want to predict how the tournament is going to play out. To do this, you simulate the various rounds. In this simulation each knight k is assigned a jousting skill level S(k). In the simulation, the knight with the higher skill level always wins the match (there will be no ties in skill level). Using this simulation predict the knights that will get to the top 3 places.

Input

The first line of input contains T, the number of test cases that follow.

The first line of each test case is the number of knights N.
The next N lines each define knights 1 through N. They have a string, which is their name, and an integer, which is their skill level, separated by a space.

Output

For each test case, output the test case number, followed by the top 3 places in the format below.

Constraints

```
1 <= T <= 8
4 <= N <= 32
N is a power of 2.
```

Example Input File

```
2
4
Lancelot 10
Galahad 8
Percival 9
Bors 5
4
Caradoc 50
Gawain 55
Lucan 72
Balin 64
```

Example Output to Screen

```
Case 1:
1: Lancelot
2: Percival
3: Galahad
Case 2:
1: Lucan
2: Gawain
3: Balin
```

Explanation of Output

In the first case, Lancelot faces Galahad, and Percival faces Bors in the round of 4. Lancelot defeats Galahad, and moves on to the finals. Percival defeats Bors and also moves on to the finals. Lancelot wins the finals, putting him in first and Percival in second. The two knights that lost in the round of 4 are Galahad and Bors, and Galahad trumps Bors in the third place match.

In the second case, Caradoc faces Gawain in the first round, and Lucan faces Balin. Gawain bests Caradoc and Lucan bests Balin. In the finals, Lucan defeats Gawain, and Balin defeats Caradoc in the third place match.

8. Meddlesome Chimpanzees

Program Name: Meddle.java   Input File: meddle.dat

Caesar, the Great Chimpanzee, has taken control of a computer and several keyboards connected to it. He has a crew of chimpanzees capable of typing up words in English on this computer. After they finish typing, Caesar wants to verify that the words were actually saved in the computer. The problem here is that a rival group of chimpanzees have secretly connected another set of keyboards to the same computer and have begun typing random letters while Caesar's crew was typing. To irritate Caesar, they have interleaved extraneous letters in Caesar's input and may have over-written a character here and there.

A word is possible if Caesar can trace the word by only reading left to right. For each word, determine whether or not the words in Caesar's modified list are possible.

Input
The first integer N denotes the number of test cases to follow. Each test case starts with a string of characters that represent the final string that was stored from the input of all the chimpanzees. The next line has an integer K. The next K lines contain K words (one word per line) that Caesar is checking. Determine if Caesar can make out the word.

Output
For every word in the input, print out that word followed by a colon and a space and a YES if the word is possible and NO if it is not. There is a blank line between each test case.

Assumptions
The words will be in lower case letters from a to z.

Example Input File
```
2
sogwimovred
4
give
words
some
me
msgqouotgalrea
4
google
moxtra
quora
square
```

Example Output to Screen
```
give: YES
words: NO
some: YES
me: YES

google: YES
quora: YES
moxtra: NO
square: YES
```

9. Mutual Friends

Program Name: Mutual.java   Input File: mutual.dat

You keep seeing this guy around in school. You remember his name, but you have no idea how you know him. You hope that maybe if you see who your mutual acquaintances are, you can figure it out. To solve this problem in true computer science fashion, you write a program to use Facebook's API, download both of your lists of friends, and check to see what friends you have in common.

Input

The first line of input contains T, the number of test cases that follow.

The first line of each test case contains one integer N, the number of friends on your friends list. The next N lines each contain the name of one of your friends. The line after that contains a single integer M, the number of that person's friends. The next M lines each contain the name of one of his friends.

Output

For each test case, print the case number on its own line, and then all of the friends you have in common (in alphabetical order), each on their own line.

Constraints

```
1 <= T <= 5
1 <= N, M <= 15
```

Example Input File

```
3
3
Tyler
Alex
Maria
3
Zach
Maria
Tyler
2
Marge
Danny
1
Michael
1
Bill
1
Bill
```

Example Output to Screen

```
Case 1:
Maria
Tyler
Case 2:
Case 3:
Bill
```

Explanation of Output

In the first case, you are both friends with Maria and Tyler. Maybe you met because Maria is in chess club and you went to that one meeting your freshman year. Or maybe you met him at one of Tyler's crazy trombone parties.
In the second case, you have no mutual friends, so you have no idea how you know this guy. Maybe he just looks like someone else you know? In case 3, you decide you really should get more friends on Facebook.

10. Paint

Program Name: Paint.java   Input File: paint.dat

You are in art class and you are bored. Your teacher is having you color different cells of a grid. Since this is a pretty mundane task, you decide to see what would happen if you were to fold the paper in half, press both sides together, take it apart and repeat the steps a number of times.
You decide to start with certain grid cells painted in, and then perform a combination of horizontal and vertical folds, and then see what the end result looks like.
However, the paint keeps seeping out of the grid lines, and you can never fold the paper perfectly, so you decide to write a program to show you the outcome.

Input

The first line of input contains T, the number of test cases that follow.
The first line of each test case contains two integers N and M, the number of rows and columns in the grid on your paper.
The next N lines each contain M characters, representing each row of the grid. At each cell, if the character is 'x', there is paint there, and if the character is '.', there is no paint there.
The line after that contains a string composed of only 'H' and 'V'. This string represents the series of folds you are going to perform to spread the paint out, in order from left to right.

Output

For each test case, output the final result of folding the paper according to the specified instructions.

Constraints

```
1 <= T <= 10
1 <= N, M <= 15
1 <= number of folds <= 10
```

Example Input File

```
3
2 4
x...
..x.
H
3 3
x.x
.x.
...
V
4 4
x...
.x..
....
....
VH
```

Example Output to Screen

```
x..x
.xx.
x.x
.x.
x.x
x..x
.xx.
.xx.
x..x
```

11. Palindromic Square

Program Name: Square.java **Input File: square.dat**

A palindromic integer reads the same forwards or backwards. All single digit integers are palindromic. Some palindromic integers are more interesting than others. For example, the palindromic integer 595 can be expressed as the sum of consecutive squares: $6^2 + 7^2 + 8^2 + 9^2 + 10^2 + 11^2 + 12^2$.

In this problem you are going to determine if a given integer is palindromic and if it is, whether it can be expressed as the sum of two or more consecutive squares.

Input
The input file will contain N integers, one integer per line ($1 < N < 20$). Each integer n on a given line will be in the range $1 < n < 1000$.

Output
For each integer you will print YES if it is palindromic and can be expressed as the sum of two or more consecutive square numbers or NO otherwise.

Sample Input
```
66
101
434
500
595
```

Sample Output
```
NO
NO
YES
NO
YES
```

12. Wordoku

Program Name: Wordoku.java　　　Input File: wordoku.dat

Wordoku is a variation of Sudoku. Instead of digits, the letters of the alphabet are used in a 9 x 9 grid. These letters usually spell out a word or phrase somewhere in the grid. The problem that you are given is a variation on Wordoku. You will be given a square grid but its dimension need not be a 9 x 9 grid. The constraints on this variation of Wordoku are slightly more relaxed. Each row and column must contain the same mix of letters. No row or column may have repeated letters. For this problem you will have to decide if a given Wordoku grid is valid or not subject to the given constraints.

Input

The first line has a single integer *N* that is the number of grids to test. The next line gives *M*, the dimension of the first grid. There are *M* lines of data, where each line has *M* lower case letters separated by one or more spaces. After the *M* lines of data is a single blank line separating one grid from another and a blank line after the last grid. There are *N* such grids. The ranges for *N* and *M* are: $1 \leq N \leq 10$ and $3 \leq M \leq 26$.

Output

There will be N lines of output. Each line will state whether the grid is VALID or INVALID.

Sample Input

```
3
3
a w l
w l a
l a w

4
g e v a
v g a e
a v e g
e a g v

5
p t a d e
a p e t d
t e d p a
d a t e p
e d o a t
```

Sample Output

```
VALID
VALID
INVALID
```