Sponsored by

Hosted by

جـــامعـــــة الشـــارقـــة
UNIVERSITY OF SHARJAH

1st Gulf Programming Contest
March 30,31 2011

University of Sharjah, UAE
Problem Set
Duration: 10 am - 3 pm

Problem #	***Problem Name***	***Balloon Color***
A	The Battle of Gondor	Purple
B	Number Maze	Red
C	Best Period Best Gain	Blue
D	Money and Coins	White
E	Quest for Perfection	Black
F	Puzzle of Letters	Green
G	-1i Representation	Yellow
H	Pineapple Pyramid	Silver
I	1-3-5-7	Pink
J	Knight's Game	Gold

Problem A:
The Battle of Gondor

Source file: gondor.{c | cpp | java}
Input file: gondor.in

The kingdom of Rensor has declared war over the state of Gondor. A number of battleships are attacking the city of Gondor. As the commander in chief of the naval force of Gondor it is your responsibility to thwart the initial wave of battleships before a full scale response. Your options are limited and you can only destroy three of the battleships in the attack. The good news is that due to the impact of destruction, all the battleships inside the area covered by the three destroyed battleships will also be destroyed. Now you must carefully study the locations of battleships and determine the maximum number of battleships that you can destroy in the attack.

Given the positions of number of battleships in an attacking fleet, your task is to find out the maximum number of battleships that can be destroyed.

Input

The first line of input contains an integer N indicating the number of test cases to follow. The first line of each test case contains a number B indicating the number of battleships, followed by B locations of battleships each consisting of two coordinates X and Y. There will be a blank line before each test case. All coordinates are 32-bit integers and the number of battleships B is in the range $3 \leq B \leq 200$.

Output

For each test case, print on one line the following message: "Maximum battleships that can be destroyed: S", where S is the total number of ships that can be destroyed.

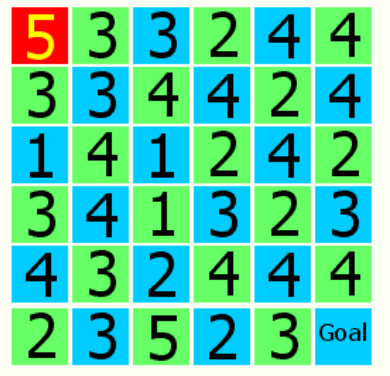
Sample Input	Output for Sample Input
2 6 0 0 5 0 3 3 -5 -4 10 -4 3 20 6 10 10 20 10 15 0 13 13 50 50 50 60	Maximum battleships that can be destroyed: 6 Maximum battleships that can be destroyed: 5

Problem B: Number Maze

Source file: maze.{c | cpp | java}
Input file: maze.in

In a number maze, a player always starts from the square at the upper left and makes a certain number of moves which will take him/her to the square marked Goal if a solution exist. The value in each cell in the maze indicates how far a player must move horizontally or vertically from its current position.

Your task is to find out if the shortest path to the cell labeled "Goal" and print it.



Input

The first line of input contains an integer N indicating the number of test cases. The first line of each test case contains two integers W and H, with W, respectively the number of columns and the number of rows of the maze. H lines follow representing the maze, each containing W integers. The goal square is indicated by the number -1 in the maze description.

Output

For each test case, output the solution of the maze or the phrase "No Solution Possible." Solutions should be output as a list of square coordinates in the format "(Column, Row)", in the order in which they are visited from the start to the goal, including the starting cell. Successive coordinates should be separated by a single space. An empty line should separate successive solutions. You will need to report the shortest solution from start to the goal. The shortest solution will be unique.

Sample Input	Output for Sample Input
3 6 6 5 3 3 2 4 4 3 3 4 4 2 4 1 4 1 2 4 2 3 4 1 3 2 3 4 3 2 4 4 4	(0 0) (5 0) (5 4) (1 4) (1 1) (4 1) (4 3) (4 5) (4 2) (0 2) (0 1) (3 1) (3 5) (5 5) (0 0) (0 1) (0 2) (0 3) (0 4) (4 4) No Solution Possible.

2 3 5 2 3 -1 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 4 1 1 3 -1 4 4 3 3 3 2 -1 2 1 2 3 3 2 1 3 1 2 1	

Problem C:
Best Period Best Gain

Source file: profit.{c | cpp | java}
Input file: profit.in

The profit (positive or negative) of the Asian Counterfeit Manufacturers company during the last n years is represented by the tuple

$$P = (p_1, p_2, \ldots, p_n).$$

For a contiguous period i..j we define the accumulated profit by

$$AP = p_i + p_{(i+1)} + \ldots + p_j.$$

The company is interested in finding the best period of its history, where the accumulated benefit has been maximal. Write a program that allows the company to find this period and the corresponding accumulated profit.

Input

The input file starts with a line holding the number N of test cases. N lines follow, each containing all the data of a test case. Each test case starts with the number M of years ($1 \leq M \leq 100$), followed by M 16-bit signed integers representing the yearly profits. Numbers are separated by whitespace.

Output

For each sequence display the accumulated profit followed by the indexes of the beginning and the end of the corresponding period. Indices start from 1. For a null or negative accumulated profit display 0 followed by -1 -1.

Sample Input	Output for Sample Input
3 7 -5 1 3 -1 10 -2 0 9 -2 2 3 4 5 4 3 2 -3 4 -2 -3 0 -1	13 2 5 23 2 8 0 -1 -1

Problem D:
Money and Coins

Source file: money.{c | cpp | java}
Input file: money.in

The central Bank of the newly formed American Union wants to issue a series of *M* coins with values $v_1, v_2, \ldots, v_M$ for its currency. In order to find the best denominations, the bank will use as a criterion the number of different coin combinations (ignoring the order) that can form a certain amount of money A. For example, for the series of coins of values 2, 3, and 5, and the amount 9, we have the following combinations:

9 = 2 + 2 + 2 + 3
9 = 2 + 2 + 5
9 = 3 + 3 + 3

To help the bank in this task, write a program that determines the **number of different combinations** (ignoring the order) that can form an amount of money A for a given series of coins $v_1, v_2, \ldots, v_M$.

Input

The input file starts with a line holding the number *N* of test cases. Each test case starts with a line holding the *A* and M values separated by a space. A single line follows with the *M* coin values. All number are separated by a single space. The amount *A* and the coin values are positive integers, with $0 \le M \le 20$ and $1 \le A \le 500$. A coin value ranges from 1 to 10.
Successive test cases are separated by an empty line.

Output

N lines listing the number of different combinations for each of the test cases. Successive answers are separated by an empty line.

Sample Input	Output for Sample Input
3 5 3 1 2 3 9 2 2 3 6 0	5 2 0

Problem E:
Quest for Perfection

Source file: perfect.{c | cpp | java}
Input file: perfect.in

As the chief of the transportation department, you are obsessed with creating a perfect road network for the country. Your idea for perfection is weird, i.e. no road should intersect another. You do not know why or how you could achieve this, but you are determined to first find out the relationship of a road's length with the number of intersections it has with the other roads. And because perfection is the ultimate goal, you are considering only straight roads!

Your task is to create a program that will output the information of the roads that intersect with at least another one with respect to their length. These roads should be sorted in a descending order according to the number of intersections they have with the other roads and their length. If for example, there are 6 roads, with the properties shown in the table below:

ID	x_1	y_1	x_2	y_2	Length	# intersections
0	0	4	12	4	12	2
1	11	2	5	5.5	6.95	4
2	10	2.5	12	2.5	2	2
3	1.1	8	1.1	0	8	1
4	10.1	0.8	10	2.8	2	2
5	4.5	3	12	3	7	1

You should first report the road with the most intersections, followed by the road that has the most intersections in the remaining roads but has a length smaller or equal to the first reported road, and so on. Thus a road with a longer length but with less number of intersections will never be reported. If two roads have same number of intersections then they will be sorted by their length in descending order. If two roads have same number of intersections and same length then they will be sorted by their IDs in ascending order. For the above set of roads your program should output the following ID, length and number of intersections:

```
1 6.95 4
2 2.00 2
4 2.00 2
```

Input

The first line of input contains an integer N indicating the number of test cases to follow. The first line of each test case contains the number R of roads, with $1<R\leq 200$. R lines follow, each containing the start (x_1, y_1) and end (x_2, y_2) coordinates of each road/line segment. The coordinates are floating point numbers.

Each road has an associated implicit ID corresponding to its position in the input (starting from 0 for the first road),. A blank line separates successive test cases.

Output

The output of the program will be the largest set of roads sorted by the number of intersections and their length in descending order. The set should start from the road with most intersections followed by the roads of lesser intersections and smaller length. If the length and number of intersections of two roads are identical then they should be sorted in ascending order of their IDs. See the problem description for more details. On each line print the ID, length and number of intersections for each road in the set. The road length should be reported with an accuracy of 2 decimal points.

There should be a blank line between the solutions of each test case.

If no road intersects any other road then print "`This is a perfect network!`"

Sample Input	Output for Sample Input
3 8 -2 8 8 8 -2 4 8 4 -2 2 8 2 -2 0 8 0 -2 -10 -2 -2 0 -10 0 -2 2 -10 2 -2 4 -10 4 3 5 -4 4 4 4 -6 3 6 3 -8 2 8 2 -10 0 10 0 0 -1 0 5 6 -1 4.5 1 4.5 -4 4 4 4 -6 3 6 3 -8 2 8 2 -10 0 10 0 -1 5 1 5	7 13.00 2 2 10.00 1 3 10.00 1 4 6.00 4 This is a perfect network!

Problem F:
Puzzle of Letters

Source file: puzzle.{c | cpp | java}
Input file: puzzle.in

A fan of jigsaw puzzles wants to design a game where groups of letters are grouped together in a puzzle-like fashion. To simplify the problem, he assumes that each group/puzzle piece is composed of two letters (e.g. EL). He assumes also that letters in the pieces cannot be swapped. Given a certain number of pieces, he wants to form a string of pieces such that the neighboring letters in each pair of consecutive pieces match. For example, if we consider the following pieces:

```
AS, ZV, AZ, KA, SK,
```

We can assemble them as follows:

```
AS SK KA AZ ZV.
```

Your task is to write a program that given the groups of letters, produces an assembly of them in the way described above. The problem conditions are as follows:

1. The number of letter groups is in the range between 1 to 100.
2. All the letters are capital.
3. If there are more that one solution, then the program should output the sequence which gives priority to the order of appearance of the pieces in the input file. For example, given this input:

```
AL CA LC
```

there are three solutions:

```
AL LC CA
CA AL LC
LC CA AL
```

The one that should be chosen is : `AL LC CA`.

Input

The first line of input contains an integer *N* indicating the number of test cases to follow. Each test case is made up of two lines: the first line contains the number *M* of pieces. The second line contains *M* pairs of letters separated by white space. A blank line separates successive test cases.

Output

For each test case you should output the sequence of pieces that joins them. If a puzzle has no solution, the output should be -1. An empty line should separate successive answers.

Sample Input	Output for Sample Input
3 3 AB XA BX 5 FG EF AB HE GH	AB BX XA -1 MN NR RA AB BC

5 MN AB NR BC RA	

Problem G:
-1i Representation

Source file: i1rep.{c | cpp | java}
Input file: i1rep.in

The *-1i* numeric representation is a complex number representation composed only of the digits 0 and 1 much like binary representation except that it is based on the powers of the complex number ***(-1 +i)***, where ***i*** is the imaginary number *(i^2 = -1)*. Therefore a number x, in this representation is expressed as follows

$$X= a_0 (-1+i)^0 + a_1(-1+i)^1 + a_2(-1+i)^2 +\ldots.$$

Where a_0, a_1 ,a_2 are digits of values 0 or 1.
From the expression above, we can see that both the real and the imaginary parts of the complex number X are integers (positive or negative). For example, the complex number -1 –i is represented as 110, since

$$-1 - i = 0 \bullet (-1+i)^0 + 1 \bullet (-1+i)^1 + 1 \bullet (-1+i)^2$$

Your task is to write a program which determines the *-1i* representation for a given complex number a +ib, where a and b are integers.

Input

The first line of input contains an integer *N* indicating the number of test cases to follow. Each test case is given on a separate line, with the integers *a* and *b* separated by a single space.

Output

For each complex number output the corresponding -1i representation on a separate line.

Sample Input	Output for Sample Input
3 -1 -1 2 1 1 3	110 1111 1010

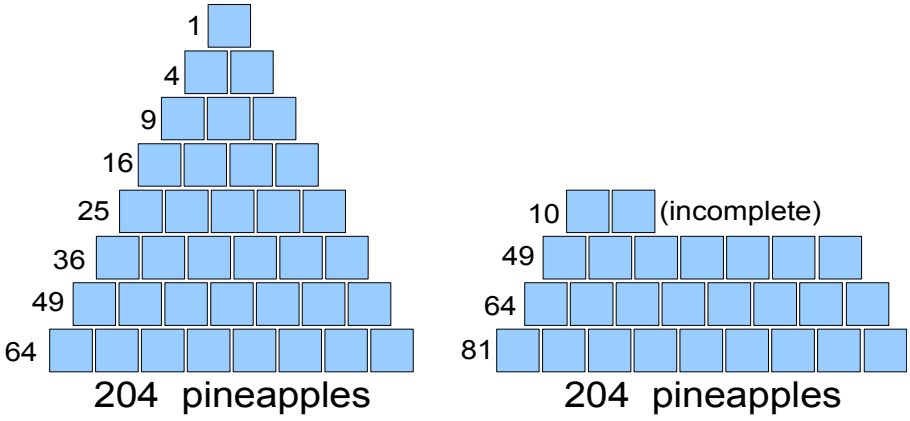
Problem H: Pineapple Pyramid

Source file: pyramid.{c | cpp | java}
Input file: pyramid.in

Gulf pineapple producers are facing a problem: a crop of a new kind of pineapple, cube sized and bound for the Japanese market, has to be stored in a closed warehouse. The pineapples are to be arranged in a pyramid with a square base. Each higher level has a side which is one pineapple smaller than the one right below it.

The warehouse has a limited height so the problem becomes the determination of the pyramid base so that all the pineapples can fit in the pyramid. For example, if the side of each pineapple cube is P=20cm and the warehouse height is H=300cm (or 3m), then storing N=1240 pineapples would require a pyramid side made of S=15 pineapples.

This solution is not unique if one is willing to use incomplete pyramids as shown in the example below:



Because several shipments need to fit in the warehouse, the problem is to find the pyramid with the smallest possible base.

Your task is to write a program that given the values of P, H and N, calculates S with the constraint described above.

Input

The input file contains a number of test cases. Each test case is on a line of each one, and contains three integers P, H and N separated by whitespace. All the numbers are 32-bit unsigned integers, with $0<P\leq H$ and $N\geq 0$. The end of the input is indicated by a line with 3 zeros. All distance measurements (P, H) are given in centimeters.

Output

For each test case you should output the side size S on a separate line.

Sample Input	Output for Sample Input
10 101 1000 10 110 1000 10 300 25000 0 0 0	15 14 43

Problem I:
1-3-5-7

Source file: beads.{c | cpp | java}
Input file: beads.in

The 1-3-5-7 game is based on a layout of 16 beads arranged in groups as shown below:

O

OOO

OOOOO

OOOOOOO

The game is played by two players who can extract during their turn as many beads (at least one) from any single group (row) as they like in one move: a group of beads can thus be split into two by having one or more consecutive beads taken from the middle of it. Beads in split groups cannot be extracted in one move. The player taking the last bead looses the game.

If for example player A is supposed to play next and the state of the game, so far, is as follows (X indicates a removed bead):

O

XXO

XXXXX

XXXXXOO

Player A wins by removing one bead from the group of two beads in the bottom row, regardless of what the second player will do next (i.e. there is a move that guarantees his victory).

Your task is to write a program which given a layout of beads and the player order (e.g. who plays next) will find which of the two players could win with certainty.

Input

The input file starts with a line containing the total number of input cases *N*. *N* lines follow; each line is made up of 6 strings separated by white space. The first four strings represent the state of the groups, where O indicates that a bead is present and X indicates that a bead was removed. The two other following strings are the names of the players. Each name is a single character long. The first of these two strings refers to the player that is playing next.

Output

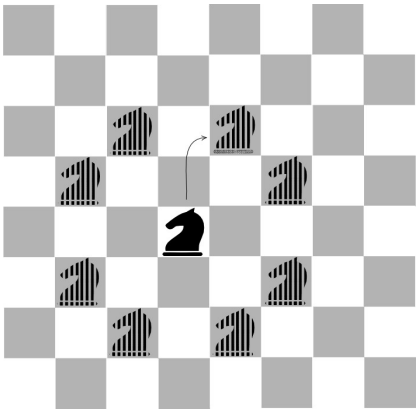
For each input case you should output the name of the player who could win with certainty.

Sample Input	Output for Sample Input
4 O XXO XXXXX XXXXXOO A B O XXO XXXXX XXXXXOO B A O XXO XXXXX XXXXXXX A B X XXX XXXXX OOOOOOO A B	A B A A

Problem J:
Knight's Game

Source file: knight.{c | cpp | java}
Input file: knight.in

The knight pieces in chess, move in an odd fashion, covering a straight or inverted gamma shape Γ. There is a claim that any layout of N chess pieces on an MxM chessboard (N is larger than or equal to M and less than M^2) can be covered in N moves by a single knight. The following figure illustrates the eight possible moves that a knight can do from a given position on the board:



Your task is to write a program that verifies or rejects this claim for arbitrary board layouts. The knight's position is also a part of the problem's input.

Input

The input file starts with a line holding the number K of test cases. Each test case starts with a line holding the size M of the board, followed by M lines describing the layout of the board; each of the lines contains M letters (K, X, or E), where 'K' indicates the position of the knight, 'X' the position of one of the N pieces to be attacked and 'E' an empty space. The size M of the board is in the range [3,8].

Output

For each test case output a 'Y' if all the positions marked by 'X' can be attacked in N moves, or 'N' otherwise.

Sample Input	Output for Sample Input
2 3 KXE EEX XEE 4 XXEE EKEE EEEE EXXE	Y N