

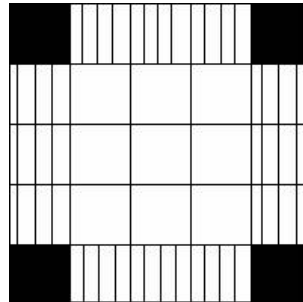
TD n°11

1 Atoms (11606)

Time limit : 2.000 seconds.

Atoms is a two player game consisting of an $N \times N$ grid. The top-left cell has a coordinate of $(0,0)$ and the bottom-right has a coordinate of $(N-1, N-1)$. Each player has an infinite number of atoms with them. Every atom of player one has the same color. The atoms of player two also have same color but they are different from those of player one. The two players take turns to place atoms on the grid. On each turn, a player selects a stable cell to place one atom from his collection. The rule for stability is simple ; the cell must either be empty or it must already contain an atom of that player's color. There is however an upper limit to the number of atoms a cell may contain and they are described below :

1. The four corner cells can contain at most 1 atom.
2. The other outer cells can contain at most 2 atoms.
3. The remaining cells can contain at most 3 atoms.



As an example, the figure above shows a grid of 5×5 . The filled cells can contain at most 1 atom each. The stripped ones can contain at most 2 atoms each and the blank ones can contain at most 3 atoms each.

When selecting a stable cell, the two players follow rather simple rules :

1. First, player one places an atom on a randomly selected cell.
2. Player two looks for a cell whose minimum distance from any cell occupied by atoms of player one is maximized. Here distances between cells are measured as Manhattan distance. The Manhattan distance between two cells is the sum of the absolute row difference and column difference. For example, the Manhattan distance between $(1,5)$ and $(4,2)$ is equal to $(4-1) + (5-2) = 3 + 3 = 6$.
3. In case of multiple cells fulfilling the criteria of rule 2, the cell with lowest row number is selected. If there is still a tie, then the cell with lowest column number is selected.
4. Player one then follows a similar strategy to that described in 2 to place another atom and the game continues likewise.

If a player is unable to make any valid move, then he is considered the loser and the game stops there.

Your task in this problem is simple. Given a state of the grid, you are to determine if this state is achievable if the players follow the rules mentioned earlier.

1.1 Input

The first line of input will start with a positive integer $T < 50$, where T denotes the number of test cases. Each case will begin with a number N ($3 \leq N \leq 7$) which denotes the size of the grid for that case. N lines will follow with N integers on each line. The absolute value of these integers will be less than 107. A positive integer means player one has placed atoms in the corresponding cell and negative integer means player two placed atoms in the cell. The number of atoms in a cell is denoted by the absolute value of the cell.

1.2 Output

For each case, output the case number first. If the given configuration is valid output “valid” output “invalid”. Look at the sample in/out for exact format.

1.3 Sample Input

```
2
3
1 0 0
0 0 0
0 0 -1
3
2 0 0
0 0 0
0 0 0
```

1.4 Output for Sample Input

```
Case 1: valid
Case 2: invalid
```

2 Creating Sudoku puzzles (10893)

Time limit : 3.000 seconds.

Sudoku is an addictive puzzle game that recently has become very popular. Given a 9×9 grid, the task is to fill all empty squares with the digits 1 to 9, so that every row, every column, and every 3×3 box contains the digits 1 through 9 exactly once. Some of the squares already have digits in them, and these must not be changed. A Sudoku puzzle always have one unique solution. The figure below to the left is an example of a Sudoku puzzle, and the figure to the right its solution.

4	2	9		5	8		3	
3	9						6	4
	1				9			
			5	1				
	4		6	2		3		
			4	8				
	7				5			
5	1						2	7
9	4	7		3	6			8

4	6	2	9	1	5	8	7	3
3	9	5	2	8	7	1	6	4
7	8	1	3	6	4	9	5	2
2	7	9	5	3	1	4	8	6
1	4	8	6	9	2	7	3	5
6	5	3	4	7	8	2	9	1
8	3	7	1	2	6	5	4	9
5	1	6	8	4	9	3	2	7
9	2	4	7	5	3	6	1	8

Now, I wouldn't want to spoil the fun of solving a Sudoku puzzle “manually” by asking you to write a program to solve an arbitrary Sudoku puzzle. Instead, you're asked to write a program which assists Sudoku puzzle designers. One way to construct new Sudoku puzzles is to make them aesthetically appealing by letting the fixed digits have nice patterns. The difficulty lies in making sure there is only one solution to the puzzle you have created. This is where a computer program can help.

Your task is, given a incomplete Sudoku puzzle (meaning that it probably has more than one solution), complete the puzzle by adding digits to the empty squares so that a puzzle with a unique solution is created. No unnecessary digits may be added. That is, if you would remove any of the added digits, a (invalid) puzzle with multiple solutions should appear. Note that this does not mean that you have to create a puzzle by adding the fewest numbers of digits (which is a much harder task).

2.1 Input

The first line in the input contains the number of test cases (at most 25). Each case consists of 9 lines, each line containing 9 characters. These lines make up the (incomplete) Sudoku puzzle. A dot ('.') is used to mark empty squares. Test cases are separated with a blank line.

2.2 Output

For each test case, output 9 lines in the same format as the input describing a Sudoku puzzle based on the input grid, satisfying the constraints above. Separate the test cases with a blank line. Any valid solution will be accepted.

2.3 Sample Input

```
2
4.29.58.3
39.....6.
..1...9..
...5.1...
.4.6.....
...4.8...
.....5..
51.....27
9.47.36.8
```

```
123.....
456.....
789.....
.....123
.....456
.....789
...123...
...456...
...789...
```

2.4 Output for Sample Input

```
4.29.58.3
39.....64
..1...9..
...5.1...
.4.6.2.3.
...4.8...
..7...5..
51.....27
9.47.36.8
```

```
123.....
4568...3.
789..1...
..5...123
3...2456
.....789
...123...
...456...
.4.789...
```

3 Cutting Cakes (11607)

Time limit : 15.000 seconds.

Alice and Bob are twins. In their birthday, they have a really large cake of dimensions $10^4 \times 10^4$. The cake has a number of flowers on it. As usual, Alice and Bob starts playing with it. First, Alice cuts the cake in two pieces, and then Bob takes the part with the maximum flowers on it. Since, Alice likes the flowers too, she will try to get maximum number of flowers. The flowers, which are incident by the cut are ruined, and thus none of them get those. The badness of a cut is the difference of the number of flowers between the two sets.

The task presented to you is not to maximize the number of flowers, Alice gets. Instead, you are given the coordinates of the flowers, and a cut, made by Alice. You need to find out the number of flowers, that are partitioned into two parts, and the number of flowers, being ruined by the cut. Alice always cuts in a straight line.

3.1 Input

First line contains T , the number of test cases. Each test case starts with an integer N , the number of flowers. Following N lines each contains two integers, x_i and y_i , the coordinate of the i -th flower. No three flowers will be collinear. After that, a line with values, $M, X_1, Y_1, X_2, Y_2, dX_1, dY_1, dX_2, dY_2$ follows. Here, M is the number of queries. Each of these queries will be a line between two given points. The end points are generated by the function given below. Each call to the function generate will produce the end points of the query line. $X_1, Y_1, X_2, Y_2, dX_1, dY_1, dX_2, dY_2$ are used to generate the lines.

function generate :

$$\begin{aligned} X_1 &= (X_1 + dX_1) \bmod 10^4 \\ Y_1 &= (Y_1 + dY_1) \bmod 10^4 \\ X_2 &= (X_2 + dX_2) \bmod 10^4 \\ Y_2 &= (Y_2 + dY_2) \bmod 10^4 \\ \text{if } X_1 &== X_2 \text{ and } Y_1 == Y_2 \text{ then} \\ Y_2 &= (Y_1 + 1) \bmod 10^4 \end{aligned}$$

P.S. : It is highly unlikely that, an $O(NM)$ solution would pass the time limits. You have to think of some more clever solution

3.2 Output

Each case starts with the line "Case #n:" where n is the number of the test case. For each line, output three integers, p, q and r ; where r is the number of flowers on the line and p and q are the number of flowers on the sides of the line.

3.3 Constraints

- $T \leq 3$
- $N \leq 1500$
- $M \leq 700000$
- For all location of flower (x_i, y_i) , $0 \leq x_i, y_i \leq 10000$. No three points will be collinear.
- $p \leq q$
- All input values will fit into a 32bit signed integer

3.4 Sample Input

```
1
4
5 5
5 10
10 5
10 10
2 -7 -7 7 7 1 1 1 1
```

3.5 Output for Sample Input

```
Case #1:
1 1 2
1 1 2
```

4 Acme Corporation (11613)

Time limit : 15.000 seconds.

Wile E. Coyote is back. He is back in the business. The business of capturing the road runner. Being the most loyal customer to the Acme Corporation, they are hoping to do some great business with him. Although, Acme makes literally every kinds of devices, all of them has a similarity, that has been kept secret for ages. All of their products use a secret element “element X” (this is kept so secret that, only you and the Acme people know about this). The decision makers of the Acme Corp. has already estimated the maximum amount of element X that can be used into manufacture every month.

For month i , the per unit manufacturing cost of “element X” is m_i , and at most n_i units can be produced. Moreover, the selling price for “element X” for that month is p_i . One more thing is that, element X older than E_i months can not be used. But good thing is that, they can store any amount of element X in their inventory (it’s the Acme Corp, they can make anything :)). Now, Acme Corporation wants to know how much element X should be produced and sold, to make the highest amount of profit.



4.1 Input

- First line contains T , the number of test cases.
- For each test case
 - First line contains two integers M and I , the number of months to consider and the cost of storing per unit of element X per month in the inventory.
 - Each of the next M lines describe the parameters for each month
 - The i th line contains 5 integers, m_i, n_i, p_i, s_i, E_i , where m_i is the per unit manufacturing cost for month i , n_i is the maximum amount that can be manufactured in this month, p_i is the selling price for that month (per unit), s_i is the maximum amount that can be sold that month, and E_i is the maximum time, element X manufactured on month i , can be stored in the inventory. For example, if for month 1, $E_1 = 3$, the elements produced in month 1 can be sold in months 1, 2, 3 and 4. But it can not be sold in month 5.

4.2 Output

For each test case, output the case number and the maximum amount of profit, Acme Corporation can make. Note that, you have to think of only M months. If any amount of element X is stored in the inventory after this period, are completely ignored. For formatting, see the sample input and output.

4.3 Constraints

- $T \leq 100$
- $M \leq 100$
- $0 \leq I, m_i, n_i, p_i, s_i \leq 10^6$
- $0 \leq E_i \leq M$

4.4 Sample Input

```
1
2 2
2 10 3 20 2
10 100 7 5 2
```

4.5 Output for Sample Input

```
Case 1: 2
```

5 Poker Hands (10315)

Time limit : 15.000 seconds.

A poker deck contains 52 cards - each card has a suit which is one of clubs, diamonds, hearts, or spades (denoted C, D, H, and S in the input data). Each card also has a value which is one of 2, 3, 4, 5, 6, 7, 8, 9, 10, jack, queen, king, ace (denoted 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A). For scoring purposes, the suits are unordered while the values are ordered as given above, with 2 being the lowest and ace the highest value.

A poker hand consists of 5 cards dealt from the deck. Poker hands are ranked by the following partial order from lowest to highest

- High Card : Hands which do not fit any higher category are ranked by the value of their highest card. If the highest cards have the same value, the hands are ranked by the next highest, and so on.
- Pair : 2 of the 5 cards in the hand have the same value. Hands which both contain a pair are ranked by the value of the cards forming the pair. If these values are the same, the hands are ranked by the values of the cards not forming the pair, in decreasing order.
- Two Pairs : The hand contains 2 different pairs. Hands which both contain 2 pairs are ranked by the value of their highest pair. Hands with the same highest pair are ranked by the value of their other pair. If these values are the same the hands are ranked by the value of the remaining card.
- Three of a Kind : Three of the cards in the hand have the same value. Hands which both contain three of a kind are ranked by the value of the 3 cards.
- Straight : Hand contains 5 cards with consecutive values. Hands which both contain a straight are ranked by their highest card.
- Flush : Hand contains 5 cards of the same suit. Hands which are both flushes are ranked using the rules for High Card.
- Full House : 3 cards of the same value, with the remaining 2 cards forming a pair. Ranked by the value of the 3 cards.
- Four of a kind : 4 cards with the same value. Ranked by the value of the 4 cards.
- Straight flush : 5 cards of the same suit with consecutive values. Ranked by the highest card in the hand.

Your job is to compare several pairs of poker hands and to indicate which, if either, has a higher rank.

5.1 Input

The input file contains several lines, each containing the designation of 10 cards : the first 5 cards are the hand for the player named "Black" and the next 5 cards are the hand for the player named "White".

5.2 Output

For each line of input, print a line containing one of the following three lines :

Black wins.
White wins.
Tie.

5.3 Sample Input

```
2H 3D 5S 9C KD 2C 3H 4S 8C AH
2H 4S 4C 2D 4H 2S 8S AS QS 3S
2H 3D 5S 9C KD 2C 3H 4S 8C KH
2H 3D 5S 9C KD 2D 3H 5C 9S KH
```

5.4 Sample Output

```
White wins.
Black wins.
Black wins.
Tie.
```