Parencedence is a brand new two-player game that is sweeping the country (that country happens to be Liechtenstein, but no matter). The game is played as follows: a computer produces an arithmetic expression made up of integer values and the binary operators ‘+’, ‘-’ and ‘*’. There are no parentheses in the expression. If Player 1 goes first he/she can put parentheses around any one operator and its two operands; the parenthesized expression is evaluated and its value is used in its place. Player 2 then does the same, and the game proceeds accordingly, Player 1 and Player 2 alternating turns. Player 1’s object is to maximize the final value, while Player 2’s object is to minimize it. A sample round might go as follows:

Initial expression: 3-6*4-7+12
Player 1’s move: 3-6*(4-7)+12    →  3-6*-3+12
Player 2’s move: (3-6)*-3+12     →  -3*-3+12
Player 1’s move: (-3*-3)+12      →  9+12
Player 2’s move: (9+12)           →  21

A game of Parencedence is played in two rounds, each using the same initial unparenthesized expression: in the first round, Player 1 goes first, and in the second, Player 2 goes first (Player 1 is always trying to maximize the result and Player 2 is always trying to minimize the result in both rounds, regardless of who goes first). Let \( r_1 \) be the result of the first round and \( r_2 \) the result of the second round. If \( r_1 > -r_2 \), then Player 1 wins; if \( r_1 < -r_2 \) then Player 2 wins; otherwise the game ends in a tie. Your job is to write a program to determine the final result assuming both players play as well as possible.

**Input**

The first line of the input file will contain an integer \( n \) indicating the number of test cases. The test cases will follow, one per line, each consisting of a positive integer \( m \leq 9 \) followed by an arithmetic expression. The value of \( m \) indicates the number of binary operators in the arithmetic expression. The only operators used will be ‘+’, ‘-’ and ‘*’. The ‘-’ operator can appear as both a unary and a binary operator. All binary operators will be surrounded by a single space on each side. There will be no space after any unary ‘-’. No combination of parentheses will ever result in an integer overflow or underflow.

**Output**

For each test case, output the case number followed by three lines. The first contains the first set of operands and operator to be parenthesized in round 1 (when Player 1 goes first) and \( r_1 \). The second line contains the analogous output for round 2. The third line contains either the phrase ‘Player 1 wins’, ‘Player 2 wins’ or ‘Tie’ depending on the values of \( r_1 \) and \( r_2 \). In the first two output lines if there is a choice between which operator should be parenthesized first, use the one which comes earliest in the original expression. Follow the format used in the examples.

**Sample Input**

2
4 3 - 6 * 4 - 7 + 12
2 45 - -67 - 3

**Sample Output**

Case 1:
Player 1 (7+12) leads to -2
Player 2 (3-6) leads to -27
Player 2 wins
Case 2:
Player 1 (-67-3) leads to 115
Player 2 (45--67) leads to 109
Player 1 wins