Do not use any built-in functions unless specified otherwise. Create a Visual Studio solution for this test. For each problem, create a console application project.

Create a console application that reads a sequence of positive integers from the input. Do not use built-in functions or arrays to implement the logic. Perform the following tasks in each iteration:

1) The iteration will terminate when the input is
   a) CTRL-Z (when the **`ReadLine()`** method returns a **`null`**) OR
   b) a zero or a negative value. 2 4 6 8 11 13 15
2) The app will sum up sequential evens or sequential odds—whichever the sequence the input values are in. Code **`number%2==0`** can be used to determine whether a number is even or not. In general, sequential evens are even numbers that are together, and vice versa.
3) When the input changes from even to odd or from odd to even OR when the iteration has terminated, the app will display the sum calculated in Task 1.
4) The app will identify the largest sum of sequential odds as well as the largest sum of sequential evens.
5) When the iteration terminates, the app will terminate and display the ***positive difference*** between the largest sum of sequential odds and the largest sum of sequential evens, calculated in Task 3. However, if there is only one type of sequence, display just that sum.

For example, the inputs 2, 4, 6, 8, 1, 3, 5, 7 have sequential evens of 2, 4, 6, 8 and sequential odds of 1, 3, 5, 7. When you enter a 1 here, the app will display The sum is 20 because the input has been changed from even (8) to odd (1). In the end, the app will display The sum is 16 and then The positive difference between the largest sums is 4 because of 2+4+6+8 – (1+3+5+7). Example:

Input	Output
791 **619** **411** **968** **697** **879** **357** **321** **864** **110** **416** **996** **384** **954** **930** **74**	**1821** **968** **2254** **4728** **2474**