Nojadera, Raymond Andrew C.

CSCI 30 – B

Pseudocode:

* First, set the value of numberOfInput based on an integer input.
* Create a loop that runs based on the value of numberOfInput and include all following codes inside the said loop.
* Set the value of outputValue to 0.
* Create a list named testCaseArray that accepts integer values and splits the inputs into individual values when a space (‘ ‘) is typed.
* When the first value inside the testCaseArray is greater than its second value, do the following:
  + Set the value of largerDigit to the first value of testCaseArray
  + Set the value of smallerDigit to the second value of testCaseArray

However, when the second value is greater than the first value, do the following instead:

* + Set the value of largerDigit to the second value of testCaseArray
  + Set the value of smallerDigit to the first value of testCaseArray
* Set the value of sum to the sum of largerDigit and smallerDigit
* Set the value of inputCount to the absolute value of largerDigit
* When the value of smallerDigit is less than 0, add the absolute value of the difference of the smallerDigit and 1 to inputCount.

However, when the value of smallerDIgit is greater or equal to 0, subtract the absolute value of the difference of the smallerDigit and 1 to inputCount instead.

* If inputCount is divided by 2 and there is a remainder 1, set the outputValue to the value of the sum of the smallerDigit and the value of the integer division of inputCount and 2.
* Add the product of the sum and the integer division of inputCount and 2 to the value of outputValue.
* Clear out the values in the list testCaseArray.
* Print “Test Case # + a: b,” substituting ‘a’ by the current loop number and ‘b’ by the value of outputValue.

Why this works:

Since each number present in the summation adds 1 to the previous value, it creates a linear pattern wherein as the number of elements increase, the value of these elements also increases. Thus, by dividing this list into two and partnering both the ends until it reaches the middle, you create a universal sum. Multiply the universal sum based on how many pairs there are, and you get the summation.

For odd numbers, the middle number is located in the list and manually added.