Lab 5 Flow of Control (II): Looping Statements

Please test the correctness of your programs in **Q-1, Q-2, Q-3 and Q-4** on **PASS**.

Q-1.
Write a program that reads numbers until -999 is entered and compute the following.

a) How many positive numbers are entered?

b) How many negative numbers are entered?

c) How many zeros are entered?

d) Sum of positive numbers

e) Sum of negative numbers

f) Average of positive numbers
Hints: *Exclude **-999** from all computations. Use **while**-loop in your program.*

Expected Output:

Example 1

```
Enter Numbers! Enter -999 to Stop:
-1
2
5
-4
2
4
0
-1
0
2
-999
Total Positive Numbers are: 5
Total Negative Numbers are: 3
Total Zeros are: 2
Sum of Positive Numbers is: 15
Sum of Negative Numbers is: -6
Average of Positive Numbers is: 3
```

Example 2

```
Enter Numbers! Enter -999 to Stop:
-999
Total Positive Numbers are: 0
Total Negative Numbers are: 0
Total Zeros are: 0
Sum of Positive Numbers is: 0
Sum of Negative Numbers is: 0
Average of Positive Numbers is: 0
```

Example 3

```
Enter Numbers! Enter -999 to Stop:
-2
-3
-4
-1
-4
0
-1
0
-2
-999
Total Positive Numbers are: 0
Total Negative Numbers are: 7
Total Zeros are: 2
Sum of Positive Numbers is: 0
Sum of Negative Numbers is: -17
Average of Positive Numbers is: 0
```

Example 4

```
Enter Numbers! Enter -999 to Stop:
-2
3
-4
1
4
0
-1
0
2
-999
Total Positive Numbers are: 4
Total Negative Numbers are: 3
Total Zeros are: 2
Sum of Positive Numbers is: 10
Sum of Negative Numbers is: -7
Average of Positive Numbers is: 2.5
```

Q-2.

Write a program to produce a square matrix with 0's down the main diagonal, 1's in the entries just above and below the main diagonal, 2's above and below that, etc.

```
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0
```

Expected Output:

Example 1

```
Enter the number of rows: 5
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0
```

Example 2

```
Enter the number of rows: 8
0 1 2 3 4 5 6 7
1 0 1 2 3 4 5 6
2 1 0 1 2 3 4 5
3 2 1 0 1 2 3 4
4 3 2 1 0 1 2 3
5 4 3 2 1 0 1 2
6 5 4 3 2 1 0 1
7 6 5 4 3 2 1 0
```

Example 3

```
Enter the number of rows: 0
Please enter positive integer.
```

Example 4

```
Enter the number of rows: 3
0 1 2
1 0 1
2 1 0
```

Hint1: *Consider using nested **for-loop**, with the outer **for-loop** responsible for each row and the inner **for-loop** responsible for each column.*

Q-3. [will be marked]

A Pythagorean triple consists of three **positive integers** a, b, and c, such that $\mathbf{a^2 + b^2 = c^2}$. Such a triple is commonly written as (a, b, c), and a well-known example is (3, 4, 5). A triangle whose sides form a Pythagorean triple is called a Pythagorean triangle, and is necessarily a right triangle. In this exercise, let's try to find the total number of possible Pythagorean triples, given a specified searching range (meaning the shortest triangle edge and the longest triangle edge are both within the range). For instance, the only solution to the searching range [3, 5] (both 3 and 5 are included) is (3, 4, 5). Hence, **we say that the number of all Pythagorean triplets between 3 (lower bound) and 5 (higher bound) is 1**.

As a straightforward solution, let's first check all possible integers in the given range. We can use a triple-nested for loop that tries all possibilities. This is an example of "brute force computing". You will learn in more advanced computer science courses that there are many interesting problems for which there is no known algorithmic approach other than using sheer brute force.

Expected Output:

Example 1

```
Please input the maximal length allowed for the edge: 10
Please input the minimal length allowed for the edge: 1
The number of all Pythagorean triplets between 1 and 10 is 2.
```

Example 2

```
Please input the maximal length allowed for the edge: 100
Please input the minimal length allowed for the edge: 10
The number of all Pythagorean triplets between 10 and 100 is 45.
```

Example 3

```
Please input the maximal length allowed for the edge: 1
Please input the minimal length allowed for the edge: 10
Invalid inputs!
```

Example 4

```
Please input the maximal length allowed for the edge: 0
Please input the minimal length allowed for the edge: -1
Invalid inputs!
```

Hint 1: *To avoid producing duplicate Pythagorean triples, start the second for loop at b = a, and the third for loop at c = b. This way, when a Pythagorean triple is found, a will be the shortest side of the triangle and c will be the longest side.*

Hint 2: *Make sure the maximal length is larger than minimal length and both are positive integers. Otherwise, output "`Invalid inputs!`".*

Hint 3: *This program can take a significant amount of time to run, for some large ranges. If you have a CPU monitor available on your system, it is worth taking a look at it when this program executes.*

Hint 4: *Do not be concerned that you are trying values that do not seem to make sense, such as a (1, 1, 500) triangle. Remember that brute-force techniques try all possible values.*

Q-4. [Challenge!]

continue …

It is easy to find out small Pythagorean triples but takes quite a lot of time to find large Pythagorean triples with the above brutal-force method. Fortunately, mathematicians have found out many other more efficient algorithmic approaches to search for Pythagorean triples. Below we show a method named the ***Dickson's method***, invented in 1920. We only need to write two for-loops with this method and can easily find large Pythagorean triples.

Dickson's method:
To find positive integer solutions to $a^2 + b^2 = c^2$, find ***positive integers*** r, s, and t such that $r^2 = 2*s*t$. Then, (a = r + s, b = r + t, c = r + s + t) form Pythagorean triples. In fact, ***all Pythagorean triples can be found by this method*** *(by setting r to every possible positive number).*
Here is an intuitive example:
Choose ***r*** = 6. Then $r^2/2$ =18. The three factor-pairs of 18 are: (1, 18), (2, 9), and (3, 6). All three factor-pairs will produce triples using the above equations.
s = 1, t = 18 gives the triple (7, 24, 25) because a = 6 + 1, b = 6 + 18, c = 6 + 1 + 18.
s = 2, t = 9 gives the triple (8, 15, 17) because a = 6 + 2, b = 6 + 9, c = 6 + 2 + 9.
s = 3, t = 6 gives the triple (9, 12, 15) because a = 6 + 3, b = 6 + 6, c = 6 + 3 + 6.

Expected Output:

Example 1
Please input the maximal length allowed for the edge: **10000** Please input the minimal length allowed for the edge: **1000** The number of all Pythagorean triplets between 1000 and 10000 is 8838.
Example 2
Please input the maximal length allowed for the edge: **50000** Please input the minimal length allowed for the edge: **10000** The number of all Pythagorean triplets between 10000 and 50000 is 34553.

The above two examples cannot be efficiently derived using the brutal-force method introduced in Q3. Please use ***Dickson's method*** to rewrite.

Hint 1: *We can iterate the **positive integer r** from the smallest possible value to the **upper bound** in the outer loop first, as **a**, **b**, and **c** are always larger than **r**. Note that the step size for iteration could be set to, for example, 2 (the intention is to make sure **r** is always even and hence $r^2/2$ is an integer). Then compute $r^2/2$ and find all possible factors of this value with an inner loop (think about how to minimize the number of iterations). Lastly, use if-condition to bound (a = r + s, b = r + t, c = r + s + t) such that **a ≥ lower bound**, **b ≥ lower bound and c ≤ upper bound**.*

Hint 2: *First try triples of small ranges as in a) to see if the results stay the same. For large triple based right-angled triangles, the maximal length in test cases will not exceed 50000, as 50000^2 reaches the limit of what **int** can represent (≤ 2^{31}-1).*