Programming Exercise

(Coverage: Loops, Accumulators, and Nested Loops)

This exercise set assumes that you already have a good understanding of counters, and loop control structures i.e., `while` loop, `for` loop, and `do-while` loop. The problems below are exercises on accumulators, and nested loops (i.e., loop within a loop). After solving the exercises CORRECTLY, you should have gained a good understanding of the following:

- ***How to properly initialize, and update the value of an accumulator variable***
- ***How to correctly specify a loop within another loop***

****

Part I. The following requires you to use accumulators. Try to solve them using the three different loop control structures.

1. Write a C program that will compute, and output the sum of all the integer numbers from 1 to 10 (inclusive).

2. Write a C program that will compute, and output the sum of all the EVEN numbers from 1 to 10 (inclusive).

3. Write a C program that will compute, and output the sum of all the ODD numbers from 1 to 10 (inclusive).

4. Solve problems (1) to (3) above, but the numbers should be generated from 10 down to 1 (i.e., in reverse sequence).

5. Write a C program that will ask the user to input an integer, say for example, n. Assume that the user enters a number that is greater than zero. Thereafter, the program should compute, and print the sum of all the integer values from 1 up to n.

6. Write a C program that will ask the user to input an integer, say for example, n. Assume that the user enters a number that is greater than zero. Thereafter, the program should compute, and print the sum of all the EVEN numbers from between 1 to n (inclusive).

7. Write a C program that will ask the user to input an integer, say for example, n. Assume that the user enters a number that is greater than zero. Thereafter, the program should compute, and print the sum of all the ODD numbers between 1 to n (inclusive).

8. Write a C program that will ask the user to input TWO integer values, say for example, m and n. Assume that m is less than n. Note that m and/or n may be positive or negative. The program should then compute, and print the sum of all integers from m to n (inclusive). For example, if m is –3 and n is 4, the sum is computed as –3 + -2 + -1 + 0 + 1 + 2 + 3 + 4 which is equivalent to 4.

9. Write a C program that will ask the user to input TWO integer values, say for example, m and n. Assume that m is less than n. Note that m and/or n may be positive or negative. The program should then compute, and print the sum of all the ODD integers between m to n (inclusive). For example, if m is –3 and n is 4, the sum is computed as –3 + -1 + 1 + 3 which is equivalent to 0.

10. Write a C program that will ask the user to input TWO integer values, say for example, m and n. Assume that m is less than n. Note that m and/or n may be positive or negative. The program should then compute, and print the sum of all the EVEN integers between m to n (inclusive). For example, if m is –3 and n is 4, the sum is computed as –2 + 0 + 2 + 4 which is equivalent to 4.

Part II. The following requires you to input several numbers which will be added using an accumulator. Try to solve them using the three different loop control structures.

11. Write a C program that will ask the user to input FIVE integer values using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute and print the sum of all the input values. An example user interaction is shown below:

```
Input a number: 8
Input a number: -5
Input a number: 0
Input a number: -3
Input a number: 16

The sum of the numbers you input is 16.
```

12. Write a C program that will ask the user to input FIVE integer values using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute and print the sum of all POSITIVE values. For example, if the user inputs are:

```
Input a number: 8
Input a number: -5
Input a number: 0
Input a number: -3
Input a number: 16

The sum of the POSITIVE numbers you input is 24.
```

13. Write a C program that will ask the user to input FIVE integer values using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute and print the sum of all NEGATIVE values. For example, if the user inputs are:

```
Input a number: 8
Input a number: -5
Input a number: 0
Input a number: -3
Input a number: 16

The sum of the NEGATIVE numbers you input is -8.
```

14. This problem requires you to combine the solutions to exercise numbers (12) and (13) above. The problem is the same, i.e., input five integers. However, instead of writing two separate programs to compute the sum of POSITIVE numbers, and the sum of NEGATIVE numbers – you are to write the codes in just one program. Hint: you will need two accumulators, one for the positive numbers, the second for the negative numbers. A sample run of the program is shown below:

```
Input a number: 8
Input a number: -5
Input a number: 0
Input a number: -3
Input a number: 16

The sum of the POSITIVE numbers you input is 24.
The sum of the NEGATIVE numbers you input is -8.
```

15. Write a C program that will ask the user to input FIVE integer values using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute and print the sum of all EVEN values. For example, if the user inputs are:

```
Input a number: 30
Input a number: -5
Input a number: 3
Input a number: 6
Input a number: -10

The sum of the EVEN numbers you input is 26.
```

16. Write a C program that will ask the user to input FIVE integer values using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute and print the sum of all ODD values. For example, if the user inputs are:

```
Input a number: 30
Input a number: -5
Input a number: 3
Input a number: 6
Input a number: -10

The sum of the ODD numbers you input is -2.
```

17. This problem requires you to combine the solutions to exercise numbers (12) and (13) above. The problem is the same, i.e., input five integers. However, instead of writing two separate programs to compute the sum of EVEN numbers, and the sum of ODD numbers – you are to write the codes in just one program. Hint: you will need two accumulators, one for the even numbers, the second for the odd numbers. A sample run of the program is shown below:

```
Input a number: 30
Input a number: -5
Input a number: 3
Input a number: 6
Input a number: -10

The sum of the EVEN numbers you input is 26.
The sum of the ODD numbers you input is -2.
```

18. Write a C program that will ask the user to input FIVE single precision floating point values (i.e. data type is `float`) using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute and print the sum of all the input values. Use the default 6 digits after the decimal point when printing the sum. An example program interaction is shown below:

```
Input a number: 6.25
Input a number: -2.12
Input a number: 3.75
Input a number: -7.894
Input a number: -0.2

The sum of the values you input is -0.214000

The sum of the POSITIVE values you input is 10.000000
The sum of the NEGATIVE values you input is -10.214000
```

19. Write a C program that will ask the user to input FIVE single precision floating point values (i.e. data type is `float`) using `scanf()` function. The input value may be negative, or positive (including zero). The program should then compute the SUM of all the POSITIVE numbers, and the SUM of all the NEGATIVE numbers. Use the default 6 digits after the decimal point when printing the sum. An example program interaction is shown below:

```
Input a number: 6.25
Input a number: -2.12
Input a number: 3.75
Input a number: -7.894
Input a number: -0.2

The sum of the POSITIVE values you input is 10.000000
The sum of the NEGATIVE values you input is -10.214000
```

Part III. The following are problems requiring you to use loops. Try to solve them using different COMBINATIONS of the three different loop control structures.

20. Write a program that will output the following:

```
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
```

21. Write a program that will output the following:

```
5
5  4
5  4  3
5  4  3  2
5  4  3  2  1
```

22. Write a program that will output the following:

```
1  2  3  4  5
1  2  3  4
1  2  3
1  2
1
```

23. Write a program that will output the following:

```
5  4  3  2  1
5  4  3  2
5  4  3
5  4
5
```

24. Write a program that will output the following:

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

25. Write a program that will output the following:

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

26. Write a program that will output the following:

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

27. Write a program that will output the following:

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

28. Write a C program that will output the tree below. Note that you will a nested loop to generate the triangular part, and another nested loop to generate the rectangular part.

```
      *
     ***
    *****
   *******
  *********
***********
    ***
    ***
    ***
    ***
```

(Note: There are more problems on the following page.)

Part IV. The following problems WERE GIVEN in previous examinations. The problems require that you have a good understanding of all the topics covered from the start of the lecture up to loops.

29. Write a program that will determine the (a) sum, (b) average, (c) lowest value, and (d) highest value from the TEN floating point values input by the user. A sample user interaction is shown below:

    Example #1:

```
Input a number: 1
Input a number: 2
Input a number: 3
Input a number: 4
Input a number: 5
Input a number: 6
Input a number: 7
Input a number: 8
Input a number: 9
Input a number: 10

Sum is 55.000000
Average is 5.500000
Lowest number is 1.000000
Highest number is 10.000000
```

    Example #2:

```
Input a number: 10
Input a number: -15
Input a number: 2
Input a number: 2.5
Input a number: -3
Input a number: -6
Input a number: 12
Input a number: 1
Input a number: 3
Input a number: 4.75

Sum is 11.250000
Average is 1.125000
Lowest number is -15.000000
Highest number is 12.000000
```

30. You were hired by ABC Bank. Your task is to write a C program to compute the interest to be applied on time deposits. The user is required to input two values, specifically: the balance, and the interest rate per annum (in percentage). The program will then compute and output the interest amount, and the new balance for the next 5 years. The new balance is rolled over the next year.

    Example interactions are shown below. The items in bold face represent the inputs from the user, the rest are outputs of the program. Your program should display the same outputs as the examples:

    Example #1:

```
Input the balance: 100000
Input the interest rate per annum (in percentage): 3.5
```

```
After 1 year(s), interest is 3500.00, new balance is 103500.00
After 2 year(s), interest is 3622.50, new balance is 107122.50
After 3 year(s), interest is 3749.29, new balance is 110871.79
After 4 year(s), interest is 3880.51, new balance is 114752.30
After 5 year(s), interest is 4016.33, new balance is 118768.64
```

Example #2:

```
Input the balance: 250000
Input the interest rate per annum (in percentage): 5

After 1 year(s), interest is 12500.00, new balance is 262500.00
After 2 year(s), interest is 13125.00, new balance is 275625.00
After 3 year(s), interest is 13781.25, new balance is 289406.25
After 4 year(s), interest is 14470.31, new balance is 303876.56
After 5 year(s), interest is 15193.83, new balance is 319070.39
```

31. You were hired by *e-FOREX,* a financial institution, to write a program that will convert monetary values one of the following modes: (i) from Philippine Pesos to US Dollars or (ii) from US dollars to their equivalent in Philippine pesos. The user is asked which type of conversion he/she wants. Assume that the current conversion rate is 1 US$ = 51.75 Pesos.

    An example of program interaction is shown below. The user inputs are shown in boldface; the rest are prompts and responses of the computer program. Note that the program should run in a loop (i.e., repetitively) until the user opts to quit the program. **Your program should follow/produce the same format.**

    Example Run:

```
Input Conversion (1) Peso to Dollar (2) Dollar to Peso (3) Quit: 1
Input amount in Philippine pesos: 1000.00
Equivalent amount in US dollars is 19.32.

Input Conversion (1) Peso to Dollar (2) Dollar to Peso (3) Quit: 2
Input amount in US dollars: 525.00
Equivalent amount in Philippine pesos is 27168.75.

Input Conversion (1) Peso to Dollar (2) Dollar to Peso (3) Quit: 3
End of the program. Have a good day!
```

    Assume that all user inputs are correct (i.e., there is no need to perform any validation). Assume also that all monetary values are real numbers.

*** END OF EXERCISE SET ***