Programming Exercise

(Coverage: Loop/Repetition Control Structure, Counter Variable)

Listed below are problems related to <u>generating (printing) numbers</u>. Try to solve the following exercises using three different loops: `while` loop, `for` loop, and `do-while` loop. Note that the exercise set is organized such that it starts with (extremely) simple problems, which then progress to more generalized problem statements. After solving the exercises CORRECTLY, you should have gained a good understanding of the following:

- ***How to use the while loop, for loop and do-while loop***
- ***How to specify logically correct initialization, conditions, and change of state in loops***
- ***How to use counter variables***
- ***How to generate a series of numbers using loops***

****

Part I. The following are very simple number generation problems. The numbers to be generated are within a pre-defined CONSTANT range (i.e., start value and end value) 0 to 10.

1. Write a C program that will generate/print the numbers from 0 to 10 (i.e., 0, 1, 2, …, 9, 8, 10). There should only be one number per line of output.

2. Write a C program that will generate the numbers from 10 down to 0, (i.e., 10, 9, 8, …, 2, 1, 0). There should only be one number per line of output.

3. Write a C program that will generate the EVEN numbers from 0 to 10 (inclusive). There should only be one number per line of output.

4. Write a C program that will generate the ODD numbers from 0 to 10 (inclusive). There should only be one number per line of output.

5. Write a C program that will generate the EVEN numbers from 10 down to 0 (inclusive). There should only be one number per line of output.

6. Write a C program that will generate the ODD numbers from 10 down to 0 (inclusive). There should only be one number per line of output.

Part II. The following are also number generation problems. The main difference with the items above is that the range of values considered is dependent on ONE VARIABLE. The value of the variable is a user input.

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 output the numbers from 0 up to n. That is, the numbers generated are: 0, 1, 2, …, n-2, n-1, n. There should only be one number per line of output.

8. 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 output the numbers from n down to 0. That is, the numbers generated are: n, n-1, n-2, …, 2, 1, 0. There should only be one number per line of output.

9. 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 output the EVEN numbers between 0 to n (inclusive). For example, if n is 9, the numbers generated are 0, 2, 4, 6 and 8. There should only be one number per line of output.

10. 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 output the ODD numbers between 0 to n (inclusive). For example, if n is 9, the numbers generated are 1, 3, 5, 7 and 9. There should only be one number per line of output.

Part III. The following are also number generation problems. The difference with the previous problems is that the range of values considered is now dependent on TWO VARIABLES. The values of the variables are user inputs.

11. 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 generate the numbers from m to n (inclusive) such that the **interval** between two consecutive numbers is 1, i.e., the numbers generated are m, m+1, m+2, …, n-2, n-1, n. For example, if m is –3 and n is 4, the numbers generated are –3, -2, -1, 0, 1, 2, 3 and 4. There should only be one number per line of output.

12. 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 generate the numbers from n down to m such that the **interval** between two consecutive numbers is 1, i.e., the numbers generated are n, n-1, n-2, …m-2, m-1, m. For example, if m is –3 and n is 4, the numbers generated are 4, 3, 2, 1, 0, -1, -2 and -3. There should only be one number per line of output.

13. 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 generate the EVEN numbers between m to n (inclusive). For example, if m is -3 and n is 4, the numbers generated are -2, 0, 2 and 4. There should only be one number per line of output.

14. 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 generate the ODD numbers between m to n (inclusive). For example, if m is -3 and n is 4, the numbers generated are -3, -1, 1 and 3. There should only be one number per line of output.

(Note: there is one more page after this)

Part IV. The following are more generalized number generating problems. The INTERVAL (also referred to as the increment/decrement) between two consecutive numbers is not necessarily fixed to 1 (or 2 as in the case of odd and even numbers series). The user will need to input THREE variables. The first two values will correspond to the range, while the last number will be the FIXED INTERVAL between two consecutive numbers to be generated in the series.

15. Write a C program that will ask the user to input THREE integer values, say for example, m, n and delta respectively. Assume that m is less than n, and delta is a positive integer greater than zero. In this problem, delta represents the interval between two consecutive numbers in the series. The program should then generate the numbers in the following manner:

    - The first number is m
    - The second number is computed as the sum of the previous number and delta. In this case, the previous number is m. Thus, the second number is m + delta.
    - The third number is computed as the sum of the previous number and delta. Since the previous number is m + delta, the third number is therefore (m + delta) + delta. This is equivalent to m + 2 * delta.
    - The fourth number is generated as the sum of the previous number and delta, and so forth…
    - The last number in the series should NOT BE GREATER than n.

    Example #1:
    Assume m is –3, n is 4, and delta is 3. The numbers generated are: –3, 0, and 3.
    Example #2:
    Assume m is 5, n is 27, and delta is 5. The numbers generated are: 5, 10, 15, 20 and 25
    Example #3:
    Assume m is –25, n is 0, and delta is 6. The numbers generated are: -25, -19, -13, -7, and -1

    There should only be one number per line of output.

16. Write a C program that will ask the user to input THREE integer values, say for example, m, n and delta respectively. Assume that m is less than n, and delta is a positive integer greater than zero. In this problem, delta represents the interval between two consecutive numbers in the series. The program should then generate the numbers in the following manner:

    - The first number is n
    - The second number is computed as the difference of the previous number and delta. In this case, the previous number is n. Thus, the second number is n - delta.
    - The third number is computed as the sum of the previous number and delta. Since the previous number is n - delta, the third number is therefore (n - delta) - delta. This is equivalent to n – 2 * delta.
    - The fourth number is generated as the difference of the previous number and delta, and so forth…
    - The last number in the series should NOT BE LESS than m.

    Example #1:
    Assume m is –3, n is 4, and delta is 3. The numbers generated are: 4, 1, and -2.
    Example #2:
    Assume m is 5, n is 27, and delta is 5. The numbers generated are: 27, 22, 17, 12, and 7
    Example #3:
    Assume m is –25, n is 0, and delta is 6. The numbers generated are: 0, -6, -12, -18, and –24

    There should only be one number per line of output.

*** END OF EXERCISE SET ***