Programming Fundamentals
CS1336

Assignment #6 – Looping 1

Assignment #6 – Looping 1

Introduction

Your sixth programming assignment will consist of two small C++ programs. Each program will be independent of the other program. Each one should compile correctly and produce the specified output.

Please note that each of the programs should comply with the commenting and formatting rules we discussed in class. For example, there should be a header for the whole program that gives the author's name, class name, date, and description. End braces should be commented, and there are alignment and indenting requirements as discussed. Please ask if you have any questions.

Program #1

Since we're now dealing with iteration logic in the labs and lecture, here are some small looping problems for practice.

For the first problem in this homework, write a single program that does the following:

a) prints out the even integers between 2 and 100.
b) prints out the integers that are multiples of 3 from 99 down to 3.
c) prints out the integers between 2 and 1,048,576 (2^{20}) that are integer powers of 2.
d) prints out the integers between 1,048,576 down to 2 that are integer powers of 2.
e) does something new involving loops, different than any of the above. (This is an exercise in being creative. Give yourself a task, and solve it.)

For each of these problems, include a counter variable inside the loop that counts how many numbers are printed out. Report that result as well.

The results for problems (a) and (b) should be printed out with 10 integers per line, right justified in a 5 byte field. For example, the output for the even integers between 2 and 40 will look like:

```
    2    4    6    8   10   12   14   16   18   20
   22   24   26   28   30   32   34   36   38   40

Number of numbers = 20.
```

Some of the numbers are fairly big in problems (c) and (d), so the output looks better if you print out 8 numbers per line instead of 10. You'll want to adjust the field size to handle the larger numbers as well.

You may use any type of loop you want, but make sure to use at least one instance of all three looping structures. Thus, if you decide to use a `for` loop for part (a), you might consider using a `while` or `do-while` loop for (b). Since there are five problems, there will obviously be two duplicate loop types among the problems. But each of the three looping structures should be represented at least once.

Note that you do not need to write separate programs for each of these problems. They should all be done within one program. However, leave a comment before each segment of code that describes exactly which problem is being solved. This is especially important for part (e).

Finally, please don't use the `pow()` function for this assignment, especially for parts (c) and (d). Part of my purpose in assigning those problems was to make the point that updating a loop control variable can involve any type of operation, not just addition and / or subtraction. If we use the `pow()` function to control the exponents in parts (c) and (d), we convert those problems into basic iteration problems and defeat that purpose.

Bonus Question

For a voluntary analysis question, worth five bonus points on the assignment, can you come up with a formula for the number of numbers printed for each of the five problems as a function of "n", the maximum number? (The formula will be different for each problem.)

As an example, a formula for the number of numbers for an algorithm that prints the even numbers between 2 and some even number "n" might be f(n) = n/2. When that is applied to n=40, as in the example above, we get 40/2 = 20 numbers, exactly as our program reveals. What about the other problems? Of particular difficulty are problems (c) and (d).

If you decide to do this part of the problem, you may submit your results as a small Word or text file along with your CPP file.

Program #2

Please implement Problem 17 on page 300 of the Gaddis text (9E) (Problem 17 on p. 296 of 8E).

Note that your program will print out one asterisk for each $100 of sales. Therefore, the number of asterisks to be printed for each store must be calculated from the number that is inputted by the user for each of the five stores. (It cannot be hard coded into the program.) In addition, your program should print out only one asterisk at a time. As a consequence, you will have to use loops to control how many asterisks are printed on a given line.

Your program should validate the user input by using an input validation loop. Make sure that the user does not enter a negative number for any of sales figures.

Problem 17, Gaddis p300 (9E)

17. Sales Bar Chart

Write a program that asks the user to enter today's sales for five stores. The program should then display a bar graph comparing each store's sales. Create each bar in the bar graph by displaying a row of asterisks. Each asterisk should represent $100 of sales.

Here is an example of the program's output.

```
Enter today's sales for store 1: 1000 [Enter]
Enter today's sales for store 2: 1200 [Enter]
Enter today's sales for store 3: 1800 [Enter]
Enter today's sales for store 4: 800 [Enter]
Enter today's sales for store 5: 1900 [Enter]

SALES BAR CHART
(Each * = $100)
Store 1: **********
Store 2: ************
Store 3: ******************
Store 4: ********
Store 5: ******************
```