Introduction to Programming
CS1336

Assignment #8 -- Functions

Assignment # 8 -- Functions

Introduction

Your eighth programming assignment will consist of two C++ programs. Your programs should compile correctly and produce the specified output.

Please note that your programs should comply with the commenting and formatting rules we discussed in class. Please see the descriptive file on eLearning for details

Program #1 – Finding Prime Numbers

This program asks you to determine a set of prime numbers, given some input from the user.

Your program should ask the user to enter an integer between 1 and 100. For purposes of discussion, let's call that number `num`. If `num` is outside that range, your program should print out an error message and re-prompt for another number (i.e., have an input validation loop).

Your program should then calculate the first `num` prime numbers and print them both to the screen ***and*** to an output file, `PrimeOut.txt`. Your output should be neatly formatted, 10 numbers per line, and should look like the following examples. For the first example, let's assume `num == 20`.

```
The first 20 primes:

    2    3    5    7   11   13   17   19   23   29
   31   37   41   43   47   53   59   61   67   71
```

or if `num == 25`:

```
The first 25 primes:

    2    3    5    7   11   13   17   19   23   29
   31   37   41   43   47   53   59   61   67   71
   73   79   83   89   97
```

Here we are using a 5 byte field just as we did in a previous assignment. Since the 100th prime is a three digit number, that should work well. (Other spacing's are obviously possible; just make sure there are 10 numbers per line.)

To solve this problem, your program should have a function called `isPrime()` that takes an integer as an argument and returns `true` if the argument is prime or `false` otherwise. (This is a `boolean` returning function as described in the book in Section 6.9.) The prototype for the function should be:

```
bool isPrime (int number);
```

Using this function can significantly simplify your main processing loop. Its use is a perfect example of *functional decomposition*, also called *modular programming*. In modular programming, we move the solution of a specific task (in this case: determining if an individual number is prime or not) to a function and then use that function to create the overall solution to our problem. Functional decomposition is an excellent programming technique and should be used in all of your programs from now on. (Note: an example of a boolean returning function and how to use it is given on p.333 of the book. There the boolean returning function is `isEven(int)`, and it is called inside an "if" statement. You can use your `isPrime(int)` function in the same way.)

For this problem, it is particularly important to develop your pseudocode <u>before</u> you start programming. Your pseudocode for the `main()` function might begin as follows:

Open file PrimeOut.txt and verify that it was opened correctly.
Get an integer `mumber` *from the user*
Verify that `number` *is between 1 and 100 inclusive*
Set `count` *= 0*
Set `currNum` *= 2*
While `count <= number`
    If currnum is prime
        ....
    End if
    .....
End While
Close PrimeOut.txt

You should develop pseudocode for the `isPrime(int)` function as well.

Please submit your `cpp` (source code) fileand your `PrimeOut.txt` files to eLearning.

Program 2 – Lowest Score Drop

For this problem, please implement Problem #11 on page 375 of Gaddis, 9th Edition. A scan of the problem is included below for those who don't have a copy of the book.

11. Lowest Score Drop

Write a program that calculates the average of a group of test scores, where the lowest score in the group is dropped. It should use the following functions:

- `void getScore()` should ask the user for a test score, store it in a reference parameter variable, and validate it. This function should be called by `main` once for each of the five scores to be entered.
- `void calcAverage()` should calculate and display the average of the four highest scores. This function should be called just once by `main` and should be passed the five scores.
- `int findLowest()` should find and return the lowest of the five scores passed to it. It should be called by `calcAverage`, which uses the function to determine which of the five scores to drop.

Input Validation: Do not accept test scores lower than 0 or higher than 100.

In general, your program should ask the user for five different test scores, determine which one is the lowest, and calculate and display the average of the remaining four highest scores. The test scores are assumed to be `int`s. Note that although the test scores in this problem are `int`s, the average should be calculated as a `double`. Have it printed out to two decimal places.

Notice that this problem includes a requirement to create three different functions besides the `main()` function. Here are some programming notes for those functions:

1) **void getScore (int &)**

The getScore() function should include a prompt to enter another test score, an input statement, and an input validation loop that makes sure the number entered is between 0 and 100 (inclusive). Once the number passes the input validation loop, the function will return the value back through a reference parameter (not the return mechanism of the function). The return data type on the function is therefore `void`.

2) **void calcAverage (int, int, int, int, int)**

This function will have five input parameters that represent the five test scores for which we are calculating the average. The function will determine the lowest of the five scores by calling the function findLowest(). Once that is determined, it will calculate the average of the remaining four scores and display that information to the screen in a nicely formatted report. (Print out an explanatory statement followed by the average value to

two decimal places.) Since it doesn't pass any data back to the calling program, the return data type on this function is `void`.

3) **int findLowest (int, int, int, int, int)**

Like calcAverage(), this function will have five input parameters. It will determine which of those five is the lowest and pass that information back through the return mechanism of the function. The return data type on this function should therefore be `int`.

Note that of the three functions described above, one function (getScore()) will pass data back to the calling function through a reference input parameter, another function (findLowest()) will pass data back to the calling function through the return mechanism of the function, and the third function (calcAverage()) only displays its results and does not pass anything back to the calling function.

There is no output file requirement in this assignment.