Programming Project 04

This assignment is worth 40 points (4% of the course grade) and must be **completed and turned in before 11:59 on Monday, February 13th .**

Assignment Overview
This assignment will give you more experience with functions and a little work with strings

Background
Basically we are going to play games with an integer number and its reversal. For example, the number 123, when reversed, is the number 321. Note that there is an edge case where a number ends in 0: 120 reversed is 21 (not 021, which is octal and the decimal number 17).

Palindromic Number
This is pretty much what you would expect. If you take an integer number, say 121, and reverse the digits you get the same value back, namely 121.

A Prime number
We already did this didn't we. A number is prime if it is only evenly divisible by itself and 1.

A Semiprime number
A semiprime number is a number that is only evenly divisible by **exactly** two other prime numbers (besides 1 and the number itself). For example, 15 (3x5) is semiprime, so is 51 (3x17). 45 is **not** semiprime (3x3x5), neither is 81 (3^4). 25 (5x5) **is** semiprime since the two prime numbers can be the same number.

An emirp number
Such a number (emirp is "prime" spelled backwards) is a number such that the number and its reverse are both prime, but the number is ***not*** a palindromic number. Thus 13 **is** emirp since 13 is prime as is 31. 101 **is not** emirp, even though it is prime, as it is a palindrome.

An emirpimes number
Such a number (emirpimes is "semiprime" spelled backwards) is a number such that the number and its reverse are both **different** semiprimes. Thus we rule out all palindromic numbers. 15 **is** emirpimes since 15 is semiprime (3x5) and 51 is semiprime (3x17).

Project Description / Specification
Reminder
Just a reminder. We provide *exactly* our function specifications: the function name, its return type, its arguments and each argument's type. The functions will be tested using the main we provide for you. If you do not follow the function specifications, these independent tests of your functions will fail. Do not change the function declarations!

Provided Main
Since we provide the main, there is a file `skeleton.cpp` that you should use to start your program. It contains the provided main, you simply add your functions to that file.

Warning!!! You have to use the provided main in your code. If you modify the provided main in any way you will receive a 0 for the project.

Functions

function: `is_prime`: return is `bool`. Argument is a single `long n`. If `n` is prime it returns `true`, otherwise it returns `false`.

function: `is_semiprime`: return is `bool`. Argument is a single `long n`. If n is semiprime, returns true, false otherwise.

function: `reverse_num`: return is `long`. Argument is a single `long n`. Returns the reversal of the provided long value

function: `is_palindrome`: return is `bool`. Argument is a single `long n`. Returns true if the number, reversed, is exactly the same as the provided n, false otherwise.

function: `is_emirp`: return is `bool`. Argument is a single `long n`. Returns true if n and its reverse are prime but n is not a palindrome. False otherwise

function: `is_emirpimes`: return is `bool`. Argument is a single `long n`. Returns true if n and its reverse are semiprime but n is not a palindrome. False otherwise.

Input and Output
We provide 6 test cases (`input1.txt, input2.txt, input3.txt, input4.txt, input5.txt, input6.txt`) that, respectively, test each of the functions. We also provide an output (`output1.txt, output2.txt, output3.txt, output4.txt, output5.txt, output4.txt`) for each of the 6 inputs so you may test your functions individually.

Format of output
Follow the format of the output exactly, as we will be writing scripts to exactly examine the output provided.

Deliverables
`proj04.cpp` -- your source code solution including the provided main (*remember to include your section, the date, project number and comments in this file*).

1) Be sure to use "proj04.cpp" for the file name (or handin might not accept it!)
2) Save a copy of your file in your CS account disk space (H drive on CSE computers). This is the only way we can check that you completed the project on time in case you have a problem with handin.
3) Electronically submit a copy of the file.

General Hints:

1. You can write as many functions are you like over and above the ones I have specified.
    a. Make sure you write the requested functions exactly as specified. They will be tested individually according to that specification.
2. All but one of the functions return Booleans, which is not very informative. Feel free to place lots of output statements in your functions so you can see what is going on.
    a. Just remember to remove the output statements before you turn in the code!!!
3. The main will take in test cases to test each function. You already have the main, so develop the functions one at a time and run the appropriate test case to see that the function works!

a. Don't write everything all at once, write one function at a time, test it and make sure it works by running the appropriate case against it.

Specific Hints

Do what you think is right, but here are some hints to help.

`is_prime`:

- You can check every number from 2 up to the n-1 and see if any of those numbers divides without remainder. If so, then it isn't prime. Otherwise it is.
    - you can be more efficient. What value do you really need to check up to (less than n-1)?
    - `break` can save you some computational time here.

`reverse_num`:

- You don't need to use a string but it sure would be easier to do so. Use the `stol` (string-to-long) and `to_string` (number-to-string) functions to convert back and forth. They are both part of the string header. Look them up in the book or on the C++ reference site listed on the 232 web page.

`is_palindrome`

- simple application of `reverse_num`

`is_semiprime`

- starting with 2 as a factor, find the prime factors of the number excluding 1 and the number itself. There should only be two!

`is_emirp` and `is_emirpimes`

- both are just combinations of the functions above