Project pp5

Programming project due week of Mon 10/14 (wk. 7)

Due dates:
Section 01 (Block H): Wed 10/16 11:00 pm
Section 02 (Block E): Thu 10/17 11:00 pm
Submissions may begin on Mon 10/14 at 5pm.

**Project pp5: 6 points. (1 additional style point.) Iterative and Recursive Functions**

This week the project has 4 parts which together add up to 6 points. Each should be submitted separately. For style we will count all parts together.

**Project pp5a: 1 point.** Test whether a given number is *prime*, using iteration.

Write a program that reads a sequence of integers and, for each, determines whether or not it is a *prime* number. The last integer in the sequence will be 0, which is the sentinel value. A prime number is an integer that is greater than 1 and has no divisors except for 1 and itself. 2, 3, 5, 7, 11, 13, 17, 19 and 23 are examples of prime numbers.

Your program must include a function that takes an integer argument and returns true if the number is prime and false otherwise. You must use iteration (any form of loop) to determine whether or not the number is prime.

The program should perform input validation to verify that the number is positive. On each number it should print one of the three possibilities: “prime”, “not prime”, “invalid number”. As in previous projects please make sure to print a $ sign before each of these. Also make sure to end the line after each output. Here is a sample run where the user’s input is given in bold:

```
Please enter a positive integer: 27
27 is $ not prime
```

```
Please enter a positive integer: -5
-5 is an $ invalid integer
```

```
Please enter a positive integer: 31
31 is $ prime
```

```
Please enter a positive integer: 0
```

**Project pp5b: 2 points.** Same as pp5a but use recursion instead of iteration.

Please redo the prime number project, but this time use recursion to determine whether or not a given number is prime.

As in pp5a your program must include a function that takes an integer argument and returns true if the number is prime and false otherwise. This function must use *recursion* to determine whether or not a that number is prime. You cannot make this entire function recursive. However, in it, you have to test whether the number has any divisors between 2 and the number – this task should be handled by a recursive function.

I.e., write a recursive function `hasdivisors(x, n)` that takes 2 integer parameters and returns true if and only if the number x has a divisor d where 2 ≤ d ≤ n. Your function that determines whether or not a given number is prime should call this `hasdivisors` function. While your program may use a loop to perform the read-until-sentinel task, it must have no other loops.

**Project pp5c: 1 point.** Print the digits of an integer in reverse order, using iteration.

Write a program that reads a sequence of integers and, for each number, prints the digits of that integer in reverse order. The program should first read the number of integers to be tested and then expect that many integers as input. It should work both for positive and negative integers.

Your program must include a function `printRev` whose input argument is an integer and this function should take care of the printing. This function must use iteration (some form of loop) to perform its task. Please use the usual $ signs for each output.
Notice that it is easy to extract the least significant digit of an integer number using the mod operation. For example if i=234 then i%10=4. You can use this fact as a basic step in extracting and printing the digits. Here is a sample run where the user’s input is given in bold:

How many integers to you want to print? 4
Please enter an integer: 12345
Here it is reversed $ 54321
Please enter an integer: -4567
Here it is reversed $ -7654
Please enter an integer: 0
Here it is reversed $ 0
Please enter an integer: 7
Here it is reversed $ 7

Project pp5d: 2 points. Same as pp5c but use recursion instead of iteration.
Please redo the reverse printing project, but this time use recursion to for the function printRev.

Programming style guidelines for this week
To get the style point, you must satisfy all of the following requirements.

- Use recursive or iterative functions as instructed.
- Produce readable output as in the examples.
- Have an introductory comment at the top of your program.
- Have a comment for each variable describing its purpose.
- Separate your program into logical sections separated by a blank line.
- Have a comment at the start of every such section.
- Choose meaningful names.

Your first submission will be used for style grading, even if your best grading score is on a different submission.

Submitting your program

Each project should be submitted separately using the appropriate project name and file name. For example assuming your program for pp5a is in a file named pp5a.cpp which is in the current directory on andante, you should submit by typing (or substitute your file name instead of pp5a.cpp) provide comp11 pp5a pp5a.cpp