CSCE 240 – Programming Assignment Two

Due: 11:59pm on Tuesday, February 18th

Program Purpose – Implement the functions described below

IsSquare – Function that takes an integer argument and returns whether or not the argument is a perfect square (is equal to an integer squared).
For example, *IsSquare(4)* should return true, and *IsSquare(5)* should return false.

IsPerfect – Function that takes an integer argument and returns whether or not the argument is a perfect number. A perfect number is a positive integer that is equal to the sum of all of its proper divisors.
For example, *IsPerfect(6)* should return true, because the proper divisors of 6 are 1, 2, and 3, and 1 + 2 + 3 = 6.
IsPerfect(12) should return false, because the proper divisors of 12 are 1, 2, 3, 4, and 6, and 1 + 2 + 3 + 4 + 6 is not equal to 12.

IsVowel – Function that takes a character and a bool as arguments. If the character argument is a vowel, the function should return true. If the character argument is not a vowel, the function should return false.
The bool argument determines whether (true) or not (false) the letter y should be considered a vowel. The bool parameter should have a default value of true.
For example, *IsVowel('y', false)* should return false.
IsVowel('y') should return true.
IsVowel('E') should return true.
IsVowel('X') should return false.
IsVowel('u', false) should return true.

IsConsonant – Function that takes a character and a bool as arguments. If the character argument is a consonant, the function should return true. If the character argument is not a consonant, the function should return false.
The bool argument determines whether (true) or not (false) the letter y should be considered a consonant. The bool parameter should have a default value of true.
For example, *IsConsonant('y', false)* should return false.
IsConsonant('y') should return true.
IsConsonant('E') should return false.
IsConsonant('X') should return true.
IsConsonant('g', false) should return true.

ToDigit – Function that takes a character argument and returns the integer equivalent of that argument if the argument is '0', '1', '2', '3', '4', '5', '6', '7', '8', or '9'. If the argument is not a digit character, the function should return 0.

Range – Function that takes two integer variables as arguments and returns the distance between the variable's values. Once the function call is complete, the variable sent for the first argument should hold the smaller of the two values, and the variable send for the second argument should hold the larger of the two value.
For example, if x = 3 and y = 7, Range(x, y) should return 4. And after the function call is complete, x will still hold 3, and y will still hold 7.
If a = 9 and b = 1, Range(a, b) should return 8. And after the function call is complete, a should hold 1, and b should hold 9.

DigitInPosition – Function that takes a double and an integer as its two arguments. The integer argument is the position of the digit in the double to be returned. The position is relative to the digit in the ones place, which is position 0. So, position -1 is 1 position to the left of the ones place, i.e. the tens place. And position 2 is 2 positions to the right of the ones place, which is the hundredths place.
For example, *DigitInPosition(18**5**.34, 0)* should return 5 since the 5 is zero positions away from the ones place.
*DigitInPosition(**1**85.34, -2)* should return 1 since the 1 is two positions to the left of the ones place.
*DigitInPosition(185.**3**4, 1)* should return 3 since the 3 is one position to the right of the ones place.

Additional Specifications

- All function prototypes must be contained in a file named program2functions.h
- All function implementations must be written in a file named program2functions.cc
- You will submit your program2functions.h and program2functions.cc files to the assignment in Blackboard.
- Programs must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).
- Be sure to review the program expectations section of the course syllabus.

Initial Testing

Initial tests for the functions are attached to the assignment in Blackboard. A makefile has been included to run your functions with the sample tests. To use the makefile, ensure that your *program2functions.h* and *program2functions.cc* files and all of the files attached to the assignment are in the same directory. Your program will be graded using this same method with modified tests.

The commands to run the sample tests are given below:

```
make testIsSquare
make testIsPerfect
make testIsVowel
make testIsConsonant
make testToDigit
make testRange
make testDigitInPosition
```

You are strongly encouraged to create additional, more rigorous tests.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compile/link of *program2functions.cc*: 1 point
IsSquare passes instructor's tests*:* 1 point
IsPerfect passes instructor's tests*:* 1 point
IsVowel passes instructor's tests*:* 1 point
IsConsonant passes instructor's tests*:* 1 point
ToDigit passes instructor's tests*:* 1 point
Range passes instructor's tests*:* 1 point
IsSquare passes instructor's tests*:* 1 point

The penalty for late assignment submissions is 10% per day up to three days after the assignment due date. No assignment submissions will be accepted more than 3 days after the due date.