CSCE 240 – Programming Assignment Two

Due: 11:59pm on Monday, September 18th

Program Purpose

Implement the numeric functions described below.

SumDigits – Function to compute the sum of the digits in the decimal representation of an integer.
SumDigits takes an integer argument and returns the sum of the digits in the absolute value of the argument.
For example, *SumDigits(123)* will return 6, and *SumDigits(-111)* will return 3

IsPalindrome – Function to determine if the decimal representation of an integer is a palindrome. Note: a palindrome reads the same forwards and backwards.
IsPalindrome takes an integer argument and returns true if the absolute value of the integer is a palindrome and false if the absolute value of the integer is not a palindrome.
For example, *IsPalindrome(98789)* will return true and *IsPalindrome(112)* will return false

CountDigitMatch – Function to determine the number of occurrences of a digit in the decimal representation of an integer.
CountDigitMatch takes two arguments, the first is an integer, and the second is non-negative one-digit integer.
The function will return the number of occurrences of the second argument in the absolute value of the first argument.
For example, *CountDigitMatch(57585, 5)* will return 3 since the digit 5 appears three times in 57585.
If the second argument is not a non-negative single-digit integer, the function will return -1.
For example, *CountDigitMatch(57585, 57)* will return -1

SameDigits – Function to determine if the decimal representation of two integers are made of exactly the same digits.
SameDigits takes two integer arguments. The function will return true if the arguments are made up of exactly the same digits, and false otherwise.
For example, *SameDigits(12981, 21189)* will return true, and *SameDigits(1131, 311)* will return false.

Factor - Function to output the prime factorization of an integer.
Factor takes one integer argument and outputs the prime factorization of the argument to the standard output device (using cout) in the following format: "argument = p_1 * p_2 * ... * p_n" where p_i <= p_j for all i < j
For example, *Factor(45)* will output "45 = 3 * 3 * 5"
If the argument is negative, the first prime factor will display as negative.
For example, *Factor(-484)* will output "-484 = -2 * 2 * 11 * 11"
If the argument is prime, the function outputs that the argument is prime. For example, *Factor(13)* will output "13 is prime"

DoubleMinToIntMinSec – Function to convert a minute value, represented as a real number, into equivalent integers for the minute and second representation of the time.
DoubleMinToIntMinSec takes three arguments: a double, and two integer variables.
The function converts the real-valued (double) minutes to integer minutes and seconds, assigning those values to the second and third arguments, respectively.
For example, after the function call

DoubleMinToIntMinSec(3.75, min, sec);

the value of min will be 3, and the value of sec will be 45.
Note: converted times are rounded to the nearest second.

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. In order 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 testSumDigits
make testIsPalindrome
make testCountDigitMatch
make testSameDigits
make testFactor
make testDoubleMinToIntMinSec
```

You are encouraged to create additional, more rigorous tests.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compile of *program2functions.cc*: 1 point
SumDigits passes instructor’s tests: 1 point
IsPalindrome passes instructor’s tests: 1 point
CountDigitMatch passes instructor’s tests: 1 point
SameDigits passes instructor’s tests: 1 point
Factor passes instructor’s tests: 2 points
DoubleMinToIntMinSec 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 that 3 days after the due date.