**Introduction To Computer Science – 150005**

**Homework Assignment #5**

**Functions**

**Comments:**

1. Use meaningful variable names
2. Comment each program (including a comment before the main program explaining its purpose and how it works). Also, at the end of each program add a comment with a sample run with its output.
3. Be careful on code readability and appearance (indentation)
4. Make sure to compute exactly what is requested in each question.
5. Whenever an input is illegal or there is an error give a message ERROR.
6. In the questions, output is marked in green and input in yellow.
7. Submit the solution according to the directions in moodle.
   1. Write a function called **area** that accepts 2 whole, non-negative numbers that represent the length and width of a rectangle and returns the area of the rectangle.
   2. Write another function also called **area** that accepts a whole, positive number that represents the radius of a circle and returns the area of the circle. Define PI to be the constant value 3.14159.
   3. Write a main program the executes the following:
      * prints the message **enter length and width of the rectangle:**
      * reads in 2 positive, whole numbers and prints the area of the rectangle using the function you wrote in a. If there was an error in the input, read in 2 new values (even if one of the numbers was ok).
      * Prints the message **enter radius of the circle:**
      * Reads in a positive whole number and prints the area of the corresponding circle using the function you wrote in b.

Sample run:

**enter length and width of the rectangle:**2 48 **enter radius of the circle:** 13.14159

Another sample run:

**enter length and width of the rectangle:**

1  
-3  
ERROR  
2  
2  
4  
**enter radius of the circle:**   
-4  
ERROR  
3  
28.2743

1. A perfect number is a number that is equal to the sum of its factors (not including itself). For example, 6 is a perfect number since its factors are 1, 2, and 3 and 6=1+2+3.
   1. Define a function called **perfect** that accepts a positive whole number and returns true if the number is perfect and false otherwise.
   2. Define another function that accepts as input a positive whole number and prints out all the perfect numbers in numeric order (separated by a space) that is less than this number. If no parameter was passed, the function prints out all perfect numbers less than 500. Use the function perfect you wrote in a.
   3. Write a main program that asks **enter a number:** reads in a number and prints out a list of all perfect numbers less than the number inputted and also a list of all perfect numbers less than 500. Use the function that you wrote in b, first passing a parameter and a second time without passing a parameter.

Sample run:  
 enter a number:

445

6 28

6 28 496

Note: If there are no perfect numbers, you should print an empty line.  
Sample run:  
 enter a number:

3

6 28 496

1. Write a program that inputs an 8 digit number that corresponds to the first 8 digits of your social security number. The program computes a check-sum of the number. You need to write the following 2 functions which is called by the main program:
   1. Function that is passed a number and returns the sum of the digits
   2. Function that is passed an id number and returns its correct check-sum.

How is the check-sum computed?

1. List all 8 digits of the id number
2. Under each digit alternatively write the weights 1 and 2
3. Multiply each digit of the id by its weight
4. replace all 2 digit numbers computed in step c with their sum
5. sum up all the digits computed in step d
6. take the units place digit of the sum computed in e
7. the check-sum is 10 minus the number found in f

Example:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Social Security # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Weights | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| Digit \* Weight | 1 | 4 | 3 | 8 | 5 | 12 | 7 | 16 |
| Sum of 2-digit numbers | 1 | 4 | 3 | 8 | 5 | 3 | 7 | 7 |

The sum of the numbers: 1+4+3+8+5+3+7+7=38

Difference between 10 and unit place: 10-8=2

Therefore check-sum is 2

Sample run:

enter your ID:

12345678

your full ID is:

123456782

1. You need to write the following 2 functions which will be called by the main program:
   1. Function called **isPrime** whichtakes a positive whole number and returns true if it is prime and false otherwise.
   2. Function called **findPrime** which takes a positive whole number and returns the smallest prime number that is greater than or equal to the number passed. You should use the function **isPrime** you wrote in a.

Write a main program that performs the following steps:

1. Writes **enter number of values:**
2. Reads in a whole number inputted by the user. If the number is negative, the program randomly chooses a number from 10-100.   
     
   Note, moodle’s automated checker does not use c++11 and therefore does not recognize nullptr. Use NULL instead. That is, srand( (unsigned)time(NULL) );
3. Prints **table size:** and the smallest prime number that is greater than or equal to the number that was read or randomly generated.

Sample run**:**

**enter number of values:**11

**table size: 11**

Another sample run:  
**enter number of values:**

25

**table size: 29**

Another Sample run:

**enter number of values:**

-1

**table size: 19**