CSC 101 Programming Assignment #2 9/21/17

Due date – Fri Oct 27th at noon

**Program 1: Discount.java**

Your program is going to calculate the price of some bags of coffee. You will first ask the user; how many bags of coffee do they want. The price per bag is $5.50 each. Your price must also include the shipping of how many boxes this shipment will require. There are three size shipping boxes available. A large box can hold 20 bags, a medium box holds 10 bags, and a small box can hold up to 5 bags. You **cannot** ship large or medium boxes that are not full. (but small ones can have 1-5 bags in them) The price of shipping per box is large = $1.80, medium = $1.00, and small = $0.60 . You also give a discount on the coffee for large quantities. Use the following chart to calculate the discount off of the coffee price (NOT off the shipping).

0-24 bags - no discount 150-199 bags - 20% discount

25-49 bags – 5% discount 200-299 bags - 25% discount

50-99 bags – 10% discount 300 and up - 30% discount

100-149 bags – 15% discount

Discount is not to be used on the boxes. Use *if* statements or a switch statement to get this done. Figuring out the number of boxes is similar to our change program. See example at bottom for what output should look like.

**Program 2: Prime.java**

Your program will search for prime numbers. You will first ask the user for the range of values to search, and use *for* loops to progress through all the numbers chosen. Note to determine if a number is a prime number, you must check to find out if there are any values besides 1 and itself that divide into it evenly. If any other numbers found then it is not prime. To check if any number is divisible, use the modulus operator which gives the value of the remainder of a division. List off all prime numbers found within a range given by the user. Do not list off numbers that are not prime. Also output at bottom the total number of primes from. See sample output at bottom of project sheet. Hint: first write the code that will determine if a given single number is prime or not, then put it in a larger loop.

**Program 3: Perfect.java**

Your program will search for perfect numbers. A perfect number is defined as a number that equals the sum of its divisors. Such as the number 6. The values 1,2, and 3 are the only divisors of 6 and they sum up to 6. Start by writing the code to pull out all of the divisors for a single number. (Again use the modulus operator to find which numbers are divisors) As you find them, print them to the screen and keep a running total (sum). Test your final total with the original number to see if it is perfect. Report to the screen if the number was perfect. Put this entire code in a loop and allow the user to try additional numbers until they decide they are done. The user can input a negative value to signal that they are finished searching.

**Program 4: Digits.java**

Your program will read in a value as an integer data type. You will then report back to the screen; how many digits your value has. For example, 23115 will report back that you have 5 digits. Your input must be read in as an integer and you must use some kind of a loop to process the number of digits. Converting to a string and then counting the size is too easy of a shortcut. Allow the program to repeat (using a loop) until the user types in a negative value.

Each of these programs must be done in a separate file. Name them **Discount.java, Prime.java,** **Perfect.java,** and **Digits.java**. Name the class in each file these names as well. If you do not name these files correctly you will lose points.

Hand in electronically – (NOT E-mail!!!)

In S-drive CSC 101 folder:

1. Create folder called **projecttwo\_firstname\_lastname**
2. Place four files named above in folder. (NOTHING ELSE)

**Example of Output for Discount.java**

Please enter the number of bags of coffee you wish to purchase: ***52***

Number of Bags Ordered: 52 - $ 286.00

Discount: 10% - $ 28.60

Boxes Used:

2 Large - $ 3.60

1 Medium - $ 1.00

1 Small - $ 0.60

Your total charge is: $ 262.60

**Example of Output for Prime.java**

Please enter in the lower bound for my search: ***12***

Please enter in the upper bound for my search: ***29***

Primes found:

13

17

19

23

29

There were 5 primes found between 12 and 29