CSE 2050 - Programming in a Second Language (C++)
Homework Assignment 3

September 2, 2014

1 Due Dates

- **September 10, by 11.59pm**

2 General information

- Input validation should be performed in all programs. Some of the questions provide hints on how to set the limits for validation.
- Submit only the source files and the associated makefile. The makefile should build all programs (i.e., multiple targets).
- Submit a single ".tar.gz" file containing the directory with all programs (and the makefile). To compress the directory, type the following command on the linux terminal:

```
tar -zcvf archive_name.tar.gz directory_to_compress
```

3 Questions

1. **Roman Numeral Converter**. Write a program that asks the user to enter a number within the range of 1 through 10. Use a switch statement to display the Roman numeral version of that number.

   Input Validation: Do not accept a number less than 1 or greater than 10.

2. **Magic Dates**. The date June 10, 1960 is special because when we write it in the following format, the month times the day equals the year:

   6/10/60

   Write a program that asks the user to enter a month (in numeric form), a day, and a two-digit year. The program should then determine whether the month times the day is equal to the year. If so, it should display a message saying the date is magic. Otherwise it should display a message saying the date is not magic.

3. **Time Calculator**. Write a program that asks the user to enter a number of seconds.

   - There are 60 seconds in a minute. If the number of seconds entered by the user is greater than or equal to 60, the program should display the number of minutes in that many seconds.
   - There are 3,600 seconds in an hour. If the number of seconds entered by the user is greater than or equal to 3,600, the program should display the number of hours in that many seconds.
   - There are 86,400 seconds in a day. If the number of seconds entered by the user is greater than or equal to 86,400, the program should display the number of days in that many seconds.

4. **Bank Charges**. A bank charges $10 per month plus the following check fees for a commercial checking account:

   $.10 each for fewer than 20 checks $.08 each for 20–39 checks

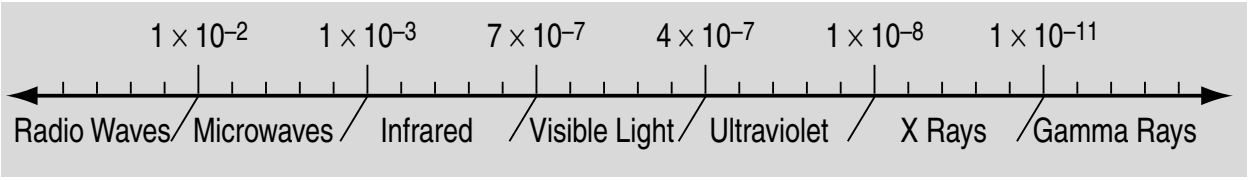
   $.06 each for 40–59 checks

   $.04 each for 60 or more checks

   The bank also charges an extra $15 if the balance of the account falls below $400 (before any check fees are applied). Write a program that asks for the beginning balance and the number of checks written. Compute and display the bank's service fees for the month.

   Input Validation: Do not accept a negative value for the number of checks written. If a negative value is given for the beginning balance, display an urgent message indicating the account is overdrawn.

5. **Spectral Analysis**. If a scientist knows the wavelength of an electromagnetic wave, she can determine what type of radiation it is. Write a program that asks for the wavelength of an electro-magnetic wave in meters and then displays what that wave is according to the chart below. (For example, a wave with a wavelength of 1E-10 meters would be an X-ray.)



6. **The Speed of Sound in Gases**. When sound travels through a gas, its speed depends primarily on the density of the medium. The less dense the medium, the faster the speed will be. The following table shows the approximate speed of sound at 0 degrees centigrade, measured in meters per second, when traveling through carbon dioxide, air, helium, and hydrogen.

Medium	Speed (Meters per Second)
Carbon Dioxide	258.0
Air	331.5
Helium	972.0
Hydrogen	1,270.0

Write a program that displays a menu allowing the user to select one of these four gases. After a selection has been made, the user should enter the number of seconds it took for the sound to travel in this medium from its source to the location at which it was detected. The program should then report how far away (in meters) the source of the sound was from the detection location.

Input Validation: Check that the user has selected one of the available menu choices. Do not accept times less than 0 seconds or more than 30 seconds.

7. **Geometry Calculator**. Write a program that displays the following menu:

```
Geometry Calculator
    1. Calculate the Area of a Circle
    2. Calculate the Area of a Rectangle
    3. Calculate the Area of a Triangle
    4. Quit
    Enter your choice (1-4):
```

   If the user enters 1, the program should ask for the radius of the circle and then display its area. Use the following formula:

$$\text{area} = \pi r^2. \tag{1}$$

   Use 3.14159 for π and the radius of the circle for r. If the user enters 2, the program should ask for the length and width of the rectangle and then display the rectangle's area. Use the following formula:

$$\text{area} = \text{length} * \text{width}. \tag{2}$$

   If the user enters 3 the program should ask for the length of the triangle's base and its height, and then display its area. Use the following formula:

$$\text{area} = \text{base} * \text{height} * .5. \tag{3}$$

   If the user enters 4, the program should end.

   Input Validation: Display an error message if the user enters a number outside the range of 1 through 4 when selecting an item from the menu. Do not accept negative values for the circle's radius, the rectangle's length or width, or the triangle's base or height.

8. **Sum of Numbers** Write a program that asks the user for a positive integer value. The program should use a loop to get the sum of all the integers from 1 up to the number entered. For example, if the user enters 50, the loop will find the sum of 1, 2, 3, 4, ... 50.

9. **Calories Burned** Running on a particular treadmill you burn 3.9 calories per minute. Write a program that uses a loop to display the number of calories burned after 10, 15, 20, 25, and 30 minutes.

10. **Long-Distance Calls**. A long-distance carrier charges the following rates for telephone calls:

Starting Time of Call	Rate per Minute
00:00–06:59	0.12
07:00–19:00	0.55
19:01–23:59	0.35

Write a program that asks for the starting time and the number of minutes of the call, and displays the charges. The program should ask for the time to be entered as a floating-point number in the form HH.MM. For example, 07:00 hours will be entered as 07.00, and 16:28 hours will be entered as 16.28.

Input Validation: The program should not accept times that are greater than 23:59. Also, no number whose last two digits are greater than 59 should be accepted. Hint: Assuming num is a floating-point variable, the following expression will give you its fractional part:

```
num - static_cast<int>(num)
```

Input Validation: Do not accept a negative starting number.

11. **Sales Bar Chart**. Write a program that asks the user to enter today's sales for five stores. The program should then display a bar graph comparing each store's sales. Create each bar in the bar graph by displaying a row of asterisks. Each asterisk should represent $100 of sales. Here is an example of the program's output.

```
Enter today's sales for store 1: 1000 [Enter]
Enter today's sales for store 2: 1200 [Enter]
Enter today's sales for store 3: 1800 [Enter]
Enter today's sales for store 4: 800 [Enter]
Enter today's sales for store 5: 1900 [Enter]
   SALES BAR CHART
   (Each * = $100)
   Store 1: **********
   Store 2: ************
   Store 3: ******************
   Store 4: ********
   Store 5: ******************
```

12. **Using Files – Numeric Processing**. The Student CD contains a file named `random.txt`. This file contains a long list of random numbers. Copy the file to your hard drive and then write a program that opens the file, reads all the numbers from the file, and calculates the following:

    A) The number of numbers in the file

    B) The sum of all the numbers in the file (a running total)

    C) The average of all the numbers in the file

    The program should display the number of numbers found in the file, the sum of the numbers, and the average of the numbers.

13. **Using Files – Population Bar Chart**. Write a program that produces a bar chart showing the population growth of Prairieville, a small town in the Midwest, at 20-year intervals during the past 100 years. The program should read in the population figures (rounded to the nearest 1,000 people) for 1900, 1920, 1940, 1960, 1980, and 2000 from a file. For each year it should display the date and a bar consisting of one asterisk for each 1,000 people. The data can be found in the `people.dat` file. Here is an example of how the chart might begin:

```
PRAIRIEVILLE POPULATION GROWTH
(each * represents 1,000 people)
1900 **
1920 ****
1940 *****
```