Western Engineering

ES1036: Programming Fundamentals – **Winter 2018**

Lab05: Modular Programming with Functions in C++

Faculty of Engineering, Western University

In order to finish this lab on time, you must start ahead of the lab session. You must try to complete the lab early and contact your TA if you require any help. The lab is due for submission on OWL at the end of your lab section.

A. Background

In this lab, we will design modular, **function-based** programs. Modular programming means that sections (modules) of the program can be used almost universally outside the code it was originally created for. Among other things, the proper design of a function usually requires careful consideration of the **input arguments** and **return values**. You will open the existing solution *ES1036Labs* and create project *Lab05*.

B. How to Get Full Credit for this Lab

You will get credit for your lab when you demonstrate it to your TA and get approval. During the code-demonstration, your TA will ask questions about your code or request you to perform some modifications. When you finish working on the lab, you will be required to upload your .cpp files (**no other file types will be accepted**) to OWL after changing the file name as indicated below. A zero grade will be awarded for a **missing OWL submission** or **missing the lab demonstration**.

- **Submission Instructions**

Number of files	:	**3**
Files to be submitted	:	.cpp source files for **E1, E2** and **E3**
Naming requirements	:	your_Western_user_name_lab05_q1.cpp,
		your_Western_user_name_lab05_q2.cpp,
		your_Western_user_name_lab05_q3.cpp.

C. Review

It is important that you review and understand the sections referenced below. Studying these sections will help you complete the lab.

Switch statement: see lecture handout Unit 4: Slides: 38 through 46. (Example on slide 44)

While loop: see lecture handout Unit 4: Slides: 54 through 69.

Do-While loop: see lecture handout Unit 4: Slides: 70 through 75. (Example on slide 72)

For loop: see lecture handout Unit 4: Slides: 77 through 80. (Examples on slides 81 - 86)

Functions: **see lecture handout 5: Slides 1 through 58.** (Examples on slides 35 – 40, 52, 56)

D. Good Programming Practice

- Learn to use the debugger. It will help you troubleshoot errors and observe variables as your code runs.
- Appropriately use functions to make your code modular. This will make your code organized and easier to understand.
- Use meaningful function names to describe the task the function performs.
- Comment your functions with a quick description of the function's task to remember why you created it and what you expect it to do.
- Use meaningful and unique variable names to avoid any scope related confusion.

E. Lab Questions

You must complete the all three of the following questions to get full marks.

E.1 Day of the Week

Requirements:

Write a program that asks a user to input their birthdate and outputs which day of the week they were born on. Additional specifications that must be followed are listed below.

Specifications:

1. You will first ask the user to input their birthdate as three separate integers (sample output shown below) and validate the integers as follows:
    a. Ensure the day entered is between 1 and 31 inclusive
    b. Ensure the month is between 1 and 12 inclusive
    c. Ensure the year is between 1900 and 2019 inclusive

2. From the main(), call a function that will return the integer result of taking the **last two digits of the year and dividing this result by 4, ignoring the remainder**. For example, if the year 1982 is entered, the function will return 20 (82/4 = 20, r=2). The function will take the user-entered year as a parameter and will return an integer and will have the following prototype:

    int findYearParameter(int year);

3. You will now **add the result returned from *findYearParameter()* to the day** entered by the user and store this result.
4. An additional integer will be **added** to the **previous step 3 result.** This number is found by calling a function that will take the month entered by the user as a parameter and return an integer as indicated below. Inside the function, you will use a switch statement or if-else logic. The function will return an integer according to this table:

Month	1	2	3	4	5	6	7	8	9	10	11	12
Return Value	1	4	4	0	2	5	0	3	6	1	4	6

    and will have the following prototype:

    int findMonthParameter(int month);

5. You must now write a third function returning a boolean value that indicates if the year entered is a leap year. There is a leap year if the year is divisible by 400 or the year is divisible by 4 but not by 100. The function will have the following prototype:
    bool isLeapYear(int year);
6. If the user-entered month is 1 or 2 (Jan or Feb) and the year entered is a leap year, subtract 1 from the **step 4 result**.
7. If the year is in the 2000's, add 6 to the **previous step 6 result.**
8. Add the last two digits of the user-entered year to the **previous step 7 result**. For example, if the year 1982 is entered, 82 will be added to the step 7 result.

9. Lastly, you will pass the final result from the previous step 8 to a function which will return a string label of the day of the week the user was born on. This will be found by dividing the parameter by 7, with the remainder being the day of the week (Or using the modulus operator). For example, 1 is Sunday, 2 is Monday etc. This function will take an integer parameter, return a string and have the following prototype:

   string findDayOfWeek(int result);

10. Display the string day of the week value from the main() to the user.
11. Note that all cin or cin statements will be in the main() function.

Sample Output:

```
//Run one

Please enter your birthdate
Day (dd): 09
Month (mm): 11
Year (yyyy): 2001
You were born on a Friday
Program ended with exit code: 0


//Run three

Please enter your birthdate
Day (dd): 24
Month (mm): 12
Year (yyyy): 1999
You were born on a Friday
Program ended with exit code: 0
```

E.2 Modular Calculator

Requirements:

Your program will present the user with a number of options corresponding to mathematical computations. The user will select an option, and execution will proceed according to the specifications. **You must not include the <cmath> or any similar math libraries.**

Specifications:

1. You will call the function *printHeader* at the start of your main function. The function will take four parameters and print a header to the console. It will not return any value. All parameters passed must appear in the printed header. You will have to write the *printHeader* function to fit the following prototype:

```
void printHeader(string name, int lab, string date, int question);
```

2. Once the console is displayed, you will call the *displayMenu* function. This function will print the calculator menu options to the console, ask the user to input a choice, validate the choice is valid menu option, and return the character choice. If the input character is invalid, keep asking for an input until a valid one is entered. You must write the *displayMenu* function to fit the following prototype:

```
char displayMenu();
```

3. You will have a switch statement in the *main* function to handle the validated return value from *displayMenu*. Each switch case will ask the user to input the appropriate value(s) and then all of the computations will be computed in separate functions. Each function will each take one or more parameters and return the computed value to the main. All results will be printed to the console from the *main*. The computation functions must have the following prototypes:

```
double computePower(double base, int exponent);
double computeSine(double radians, int numberOfIterations);
double computeExponential(double exponential, int numberOfIterations);
double computeFactorial(int number);
```

4. The power function case must proceed by asking the user to enter a real number for the base and an integer for the raised power. The raised power can be both positive or negative, but you can assume the user will enter an integer. **Hint:** A negative exponent of 4, for example, is equivalent to 4 divisions by the same base value. While a positive exponent of 4 is equivalent to 4 multiplications by the same base value.

5. The sine function case will ask the user to enter a real value in radians and compute the sine of the value according to the formula below. You can assume the user will input a real number in radians. **Hint:** You will have to ask the user to enter a value for **N**. The larger this value, the more accurate your approximation will be but a large value will slow down computation significantly. Experiment with different inputs.

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \approx \sum_{n=0}^{N} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$$

6. The exponential function case will ask the user to enter a positive or negative real value and compute the exponential of the value according to the formula below. You can assume the user will input a positive or negative real number. Ask the user to input a value for N.

$$exp(x) = \frac{x^0}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + \cdots \approx \sum_{n=0}^{N} \frac{x^n}{n!}$$

7. The factorial function will ask the user to input an integer and you can assume they will. You must validate that the integer is positive. **This factorial calculation must be implemented in a for loop.**

8. You must remove duplication of factorial and power computations. For example, when computing the sine approximation, you must compute a factorial. This should be done by calling your *computeFactorial* function and not by re-writing the same code in multiple places. Both the sine approximation and the exponential approximation will use your *computeFactorial* and *computePower* functions. **You should separate and test the functions for computing factorials and powers first before moving on to the rest.**

9. **There will be no cout or cin statements in the functions: *computePower, computeSine, computeExponential, computeFactorial*.** For these functions, you will be obtaining the required values in the main, passing them to the functions, and returning the computed values back to the *main* to be displayed.

10. Once the user has chosen to exit the program, you will call the *printFooter* function. This function will print a goodbye message to the console but not return a value or take any parameters. The *printFooter* function must be written according to the following prototype:

```
void printFooter();
```

Sample Output:

```
------------------------------
Student Name
Lab #5
Date:
Question #2
------------------------------

---------------------------
-- Calculator Application --
---------------------------
 a.  Power Function
 b.  Sine Function
 c.  Exponential Function
 d.  Factorial Function
 e.  Exit
---------------------------
Please enter your choice: a
Please enter a real number base: 4.3
Please enter an integer exponent: 5
The result is: 1470.08
---------------------------
-- Calculator Application --
---------------------------
 a.  Power Function
 b.  Sine Function
 c.  Exponential Function
 d.  Factorial Function
 e.  Exit
---------------------------
Please enter your choice: b
Please enter a number in radians: 0.5
Please enter the number of iterations: 15
The result is: 0.479426
---------------------------
-- Calculator Application --
---------------------------
 a.  Power Function
 b.  Sine Function
 c.  Exponential Function
 d.  Factorial Function
 e.  Exit
---------------------------
Please enter your choice: c
Please enter a number for the exponential: 8.5
Please enter the number of iterations: 5
The result is: 735.236
```

Sample Output Continued:

```
----------------------------
-- Calculator Application --
----------------------------
 a.  Power Function
 b.  Sine Function
 c.  Exponential Function
 d.  Factorial Function
 e.  Exit
----------------------------
Please enter your choice: c
Please enter a number for the exponential: 8.5
Please enter the number of iterations: 25
The result is: 4914.76
----------------------------
-- Calculator Application --
----------------------------
 a.  Power Function
 b.  Sine Function
 c.  Exponential Function
 d.  Factorial Function
 e.  Exit
----------------------------
Please enter your choice: d
Please input a positive integer: 9
The result is: 362880
----------------------------
-- Calculator Application --
----------------------------
 a.  Power Function
 b.  Sine Function
 c.  Exponential Function
 d.  Factorial Function
 e.  Exit
----------------------------
Please enter your choice: e
-----------------------------
Goodbye!
-----------------------------
```

E.3 Dice Game Using rand()

Requirements:

Write a dice rolling program to roll a die to reach a total dice value of 25.

Specifications:

1. For this question, you will be simulating the rolling of dice. This can be accomplished by using the built in *rand* function discussed in lecture unit 5, slides 50 through 58. You will use this function to randomly generate a number between 1 and 6 inclusive to simulate a single roll of a die.
2. Create a function *rollDice* that returns an integer result of the rolled dice. The function will have the following prototype:

   ```
   int rollDice();
   ```

3. Inside your main function, you will tell the user the goal score of the game (25) and tell them their current score (starts at 0 and gets larger the more dice they roll).
4. Ask the user to choose if they would like to roll another die or stay at their current score.
5. If they wish to roll, call the *rollDice* function and add the returned result to the user's current score. You should also display the result returned from the *rollDice* function in the main with a print statement. For example,

   ```
   cout << "The result of the current roll is: " << result;
   ```

   See the sample output below for a more detailed explanation of required outputs.
6. Your game will loop until the user chooses to stop rolling or their score has exceeded 25.
7. When the main game loop ends (user chooses to stop or 25 score is exceeded):
   1. If their score is under 25, print their score.
   2. If their score is greater than 25, print a message indicating the user rolled too many.

Sample Output:

```
The goal is a score of: 25
Your current running score is: 0
Would you like to roll the die? (y/n): h
The goal is a score of: 25
Your current running score is: 0
Would you like to roll the die? (y/n): y
The result of the current roll is: 6

The goal is a score of: 25
Your current running score is: 6
Would you like to roll the die? (y/n): y
The result of the current roll is: 4

The goal is a score of: 25
Your current running score is: 10
Would you like to roll the die? (y/n): y
The result of the current roll is: 6

The goal is a score of: 25
Your current running score is: 16
Would you like to roll the die? (y/n): y
The result of the current roll is: 2

The goal is a score of: 25
Your current running score is: 18
Would you like to roll the die? (y/n): y
The result of the current roll is: 3

The goal is a score of: 25
Your current running score is: 21
Would you like to roll the die? (y/n): y
The result of the current roll is: 1

The goal is a score of: 25
Your current running score is: 22
Would you like to roll the die? (y/n): y
The result of the current roll is: 3

The goal is a score of: 25
Your current running score is: 25
Would you like to roll the die? (y/n): n
Game over!
Your score was :25
Program ended with exit code: 0
```

F. Additional questions for practice

1. A particular talent competition has 5 judges, each of whom awards a score between 0 and 10 to each performer. Fractional scores, such as 8.3, are allowed. A performer's final score is determined by dropping the highest and lowest score received, then averaging the 3 remaining scores.

Requirements:

Write a program that uses the above rules to calculate and display a contestant's score. It should include the following functions:

- getJudgeScore should ask the user for a judge's score and validate it to be between 0 and 10. This function should be called from main() for each of the 5 judges.
- calcScore should calculate and return the average of the 3 scores that remain after dropping the highest and lowest scores. This function should be called from main, with 5 different scores as argument.
- Implement two other functions such as findLowest and findHighest to find the lowest and highest scores of the 5 scores passed to them. These two functions should be called from the calcScore function.
- Implement the code in a way so that the user gets an option to repeat the above method or stop it.