Introduction

This week's lab involves taking several concepts you've learned over the course of the semester to create a program that converts a Roman numeral to its respective decimal base 10 value.

Lab Objectives

By successfully completing today's lab, you will be able to:

- Convert ASCII character representation of roman numerals to decimal values
- Use parallel arrays to represent data
- Use functions to implement the operations

Prior to Lab

- This lab corresponds to zyBooks chapters 4, 5 & 6.

Deadline and Collaboration Policy

- Part 2 of this assignment is due by 11:59 PM on Tuesday (10/29/2019) via Canvas.
- Part 3 of this assignment is due by 11:59 PM on Friday (11/1/2019) via Canvas.
    - More instructions on what and how to submit are included at the end of this document.
- You should write your solutions to Part 2 and Part 3 of this lab by yourself. In this lab, you will talk at a high level with others in your lab about a solution. However, you will create all code by yourself and **you should not talk about specific code to anyone but a course instructor or lab teaching assistant.**
- Your zyBooks chapters and lecture slides are available resources you can use to assist you with this lab.
- For this lab, when you talk with a TA for help, you must present your planning document! You should be able to use your planning document to help explain your code and what you're working on and having trouble with. This should hopefully make the process easier and give you better feedback.

Lab Instructions

Today's lab involves creating a program which converts a list of characters (Roman numerals) to a positive integer value in base 10. A Roman Numeral is a sequence of the following characters as shown in the table and each Roman Numeral has s decimal value as shown:

Roman numeral	Base 10 value
M	1000
D	500
C	100
L	50
X	10
V	5
I	1

Roman Numerals are always written from larger values to smaller values, for example:

M = 1000
MM = 1000 + 1000 = 2000
MDX = 1000 + 500 + 10 = 1510
III = 1 + 1 + 1 = 3

The decimal equivalent is computed by adding up the decimal value of the Roman Numerals. There is one exception to this with one exception: **if the numeral is smaller than the next value, that value is subtracted.** For example:

MCM = 1000 + (-100) + 1000 = 1900 (since C is smaller than M)
LIX = 50 + (-1) + 10 = 59 (since I is smaller than X)

You program must prompt the user for a Roman Numeral as a string, convert the numerals to their respective decimal value, then compute the resulting decimal value.

You should test your program with the following inputs:

MCXIV = 1114
CCCLIX = 359
MXVI = 1016
MDCLXVI = 166

Your program must loop accepting Roman Numerals (up to 20 characters long), converting them to their decimal equivalent, and printing the results until the user enters EXIT (which is not a valid Roman Numeral and should not be converted).

Your program must validate that only legal Roman Numerals have been entered (ie, only the uppercase characters listed above are valid roman numerals) and that the Roman Numeral is not too long.

Your program must contain

- A typedef with struct called `romanNumeral` that holds the
  - The roman numeral as a string
  - The decimal value of the roman number as an integer
- Implementations of the function prototypes shown here (starter code is in Canvas)
  NOTE: you should not need to change the main function in the starter code to complete this lab.

```
/*
       This function prompts the user for a Roman Numeral
       and calls ProperRoman to validate the input until
       a valid Roman Numeral is entered or the user types EXIT
       to end the program.

       When the Roman Numeral is valid, the function returns true.

       When EXIT is entered, the function returns false.
*/
```

```
bool Prompt(char romanNumeral[], int maxSize);
/*
       This function takes a Roman Numeral and validates that only
       legal digits have been entered.

       The function returns true when a valid Roman Numeral is entered, otherwise
       it returns false.
*/
bool ProperRoman(const char romanNumeral[]);
/*
       This Function takes a valid Roman Numeral struct, calculates the integer
       equivalent, and returns the updated struct.
*/
romanNumeral ConvertRoman(romanNumeral romanNumeralToConvert);
/*
       This funciton takes a valid Roman Numeral struct and prints out the roman number
       and integer value nicely formatted as:
       XX = 20
*/
void PrintRoman(romanNumeral romanNumeralToConvert);
```

Sample Output:

```
./a.out


Please input a value in Roman numerals or EXIT to quit: MM
MM = 2000

Please input a value in Roman numerals or EXIT to quit:
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
Error! Too many characters!! Roman Number must be less than 21 characters long.

Please input a value in Roman numerals or EXIT to quit:  XYZ
Illegal Characters.

Please input a value in Roman numerals or EXIT to quit: MXVI
MXVI = 1016

Please input a value in Roman numerals or EXIT to quit: IX
IX = 9

Please input a value in Roman numerals or EXIT to quit: LC
LC = 50

Please input a value in Roman numerals or EXIT to quit: EXIT
Thank you for using the Roman Numeral to Decimal calculator.
Goodbye!
```

Lab Exercise

Create a lab9.c file in your Lab 9 directory. Your file should meet the following criteria:

- Function prototype and implementations for
  - *Struct for the roman numeral*
  - *Prompt*
  - *ProperRoman*
  - *ConvertRoman*
  - *PrintRoman*

- Your main program must
  - use these functions to implement the solution
  - loop until EXIT is entered

Bonus:

- Create a secondary program called lab9_bonus.c. This should meet the same specifications as lab9.c, but instead of converting Roman Numerals to Decimal, convert Decimals to Roman Numerals in the fewest characters possible.
- The output is shown below. Limit your range of decimal numbers to 1-5000.
- The program must exist when the user types a -1.
- https://www.math-only-math.com/rules-for-formation-of-roman-numerals.html

Sample Output:

```
./a.out
Please input a positive Decimal Number or -1 to quit: 2000
2000 = MM

Please input a positive Decimal Number or -1 to quit: 5000000
Error! Number must be less than 5000.

Please input a positive Decimal Number or -1 to quit: 2016
2016 = MMXVI

Please input a positive Decimal Number or -1 to quit: -99
Illegal input.

Please input a positive Decimal Number or -1 to quit: -1
Thank you for using Decimal Number to Roman Numeral calculator.
Goodbye!
```

Lab 08 Structure	
Tuesday	Thursday
30 minutes: work with 1-2 other individuals on Part 1 20 minutes: work individually on Part 2 Submit results of Part 2 to Canvas by 11:59PM.	50 minutes: work individually on Part 3 Submit to Canvas by 11:59PM on Friday.

Part 1: Planning with a Group (First 30 minutes)
***** Do not use a computer or calculator in this part *****

Working with the people in your lab session, brainstorm ways that you can write a program that implements the problem. Specific items that you should consider with your group during your planning session:

- Verify that you each of you knows how to manually convert between Roman Numerals to decimal numbers.
- Draw out a flow chart that shows the steps that need to happen
- Decide on any sentinel values for loops (e.g., when do loops finish).

Part 2: Planning on Your Own (20 minutes of Tuesday Lab)
***** It's okay to use a computer and/or calculator for this portion *****

Note: This should not be when you're coding your solution. Make sure that you follow the steps below, using the guidelines and sample code above, to write and then submit your document before you begin.

Taking the information that was created during your brainstorm, write your own solution *in psuedocode*. How you choose to structure the planning document is of your choice, but you need to do the following:

- Review the explanation of *pseudocode* in the Safari playlist for this lab (https://learning.oreilly.com/playlists/1480ac0b-84d1-4ef6-97f4-f6936af15daf) or using the pseudocode.pdf document in Canvas.
- *Pseudocode* is NOT code. It is structured English that where you convey the key parts of your algorithm.
- Thoroughly consider all the issues you need to fully create a program as specified above (such as how to make the formatting look correct, how many variables do you need, what type, etc.).
- Include the following mandatory header:
    - Your First (or preferred name) and Last Name
    - Your Lab and Lecture Section
    - Lab 09 – Part 2
    - Today's Date
    - Collaboration Statement: In this statement you indicate who you worked with, and which lecture sections they are enrolled in.

Submit this part of the assignment to Canvas by 11:59PM (as specified in the submission Guidelines). Late submissions will not be accepted. This part is to help you think before you code.

Part 3: Implementation (50 minutes on Thursday).

Grading Rubric

- If you are not present and attending lab during Part 1 and Part 2 of this lab on Tuesday (10/29/2019), you will not receive credit for the planning portion of this lab.
- If your document for Part 2 is not submitted successfully to Canvas by the due date (11/1/2019), you will not receive credit for the planning portion of this lab.
- If you do not check in your code with a TA for Part 3 of this lab on Thursday, you will not receive credit for the coding portion of this lab!
- If your code for Part 3 is not submitted successfully to Canvas by the due date, you will not receive credit for the coding portion of this lab.
- Your assignment will be graded out of 100 points. The approximate grading distribution is:
    - Brainstorming document completeness/accurate
        - Logic covers all requirements specified — 10 points
        - Document is simple and easy to follow — 10 points
    - Program lab9.c
        - Program operates correctly (and all functions implemented) — 25 points
            - *Struct for the roman number* — 5 points
            - *Prompt* — 5 points
            - *ProperRoman* — 5 points
            - *ConvertRoman* — 5 points
            - *PrintRoman* — 5 points
        - Output is formatted correctly — 10 points
    - Program compiles without errors or warnings — 10 points
    - Proper code formatting and commenting (See Code Style Guide on Canvas) — 15 points
    - Optional bonus exercise lab9_bonus.c — up to + 10 bonus points
        - Requires lab9_bonus.c is submitted