CS215: Introduction to Program Design, Abstraction and Problem Solving
(Spring, 2025)
Lab Assignment 5
(20 points)
Today's Date: Tuesday, February 11
Demonstration Due Date: the end of Lab6 class
Submission Due Date: Friday, February 21

The purpose of this lab assignment is

- to practice using nested loop structure
- to practice defining functions; calling other functions
- to get familiar with the user input validation

Problem Statement

Create an empty project, named **Lab5**, in Visual Studio IDE, and add a new source file named **Lab5.cpp** into the project. Then copy and paste the source file, named **Lab5.cpp**, from the following link:
https://www.cs.uky.edu/~yipike/CS215/Lab5.cpp
Complete the definitions of **THREE** functions declared before the main function in **Lab5.cpp**:

- Define a function **string roman_numeral(int n)** that converts n into Roman Numeral and returns a string form of this Roman Numeral, where n must be in the range of **[1, 3999]**. The rules to form ***Roman Number System*** are described in ***Programming Project 3.21.13 of CS215 Zybook***. For the definition of this function, you are required to call another function, named **roman_digit()**, which has been given to you in **Lab5.cpp**. (Hints: how to call **roman_digit()** function to form the corresponding ***Roman Numerals*** are described in ***Programming Project 5.22.17 of CS215 Zybook***.)

  (Please note that you are NOT allowed to use any data structures such as arrays, vectors, …and so on to define this function. You may check the solution from ChatGPT, which can be found at the last page of this document for your future reference, however any similar solution from ChapGPT version CANNOT be accepted for this Lab assignment!!!)

Roman Numbers. (The following description is from ZyBook Programming Project 3.21.13). The Roman number system has digits

I	1
V	5
X	10
L	50
C	100
D	500
M	1,000

Numbers are formed according to the following rules:

(1) Only numbers up to 3, 999 are represented.

(2) As in the decimal system, the thousands, hundreds, tens, and ones are expressed separately.

(3) The numbers 1 to 9 are expressed as

I	1
II	2
III	3
IV	4
V	5
VI	6
VII	7
VIII	8
IX	9

As you can see, an I preceding a V or X is subtracted from the value, and you can never have more than three I’s in a row.

(4) Tens and hundreds are done the same way, except that the letters X, L, C and C, D, M are used instead of I, V, X, respectively.

- Define a function **`void printTri(int n)`** that prints a triangle with the total line number equal to **`n`**. The triangle contains one little star (asterisk symbol) at the first line, three little stars at the second line, and so on till **`2*n-1`** little stars at the n^{th} line, and it is symmetric. For example, the triangle with size 5 are shown in the following:

```
    *
   ***
  *****
 *******
*********
```

- Define a function **`void printTriR90(int n)`** that prints a triangle obtained after rotating 90 degrees clockwise from the original one by calling **`printTri(n)`**. When size = 5, for instance, functions `printTri(), printTriR90(),` will print:

  //original:

```
    *
   ***
  *****
 *******
*********
```

  //after 90 degree rotation clockwise

```
*
**
***
****
*****
****
***
**
*
```

- Complete the block of code in the main function (at line number 50 of **`Lab5.cpp`**), so that your program will display the sequence of numbers: 2^0, 2^1, 2^2, …, 2^{12} and their Roman Numerals, one number at each line. (Please use the constants defined in the main function instead of "Magic Numbers", and

later you only need to modify one constant variable value to recompile your program during the demonstration)

After you complete the definitions of the above three functions, together with the block of code in the main function (at line number 50), you can start to test running your program.

The following shows the sample output of running your program: (Note that the blue part represents the user input, and "↲" represents the return/enter key)

```
Welcome to CS215 Roman Numeral Converter!
Decimal          Roman Numerals

  1       -->              I
  2       -->             II
  4       -->             IV
  8       -->           VIII
 16       -->            XVI
 32       -->          XXXII
 64       -->           LXIV
128       -->         CXXVIII
256       -->           CCLVI
512       -->            DXII
1024      -->           MXXIV
2048      -->        MMXLVIII
4096      -->     Error: NOT in the correct range!


Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: size 5 is my choice↲
Invalid size! Expecting an integer in [1, 50]
Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: five as the size↲
Invalid size! Expecting an integer in [1, 50]
Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: 5 as the size↲
The triangle with size 5 (ROMAN NUMBER: V ) is:
    *
   ***
  *****
 *******
*********
The rotation for 90 degrees clockwise:
*
**
***
****
*****
```

```
****
***
**
*


Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: 91 is that ok?↲
The size is not in the correct range! Expecting the size in
[1, 50]
Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: -5 is my guess↲
The size is not in the correct range! Expecting the size in
[1, 50]
Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: 0↲
The size is not in the correct range! Expecting the size in
[1, 50]
Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: 1 works?↲
The triangle with size 1 (ROMAN NUMBER: I ) is:
*
The rotation for 90 degrees clockwise:
*


Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: 7.5 is the size↲
The triangle with size 7 (ROMAN NUMBER: VII ) is:
      *
     ***
    *****
   *******
  *********
 ***********
*************
The rotation for 90 degrees clockwise:
*
**
***
****
*****
******
*******
******
*****
****
```

```
***
**
*


Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: 13 thirteen↲
The triangle with size 13 (ROMAN NUMBER: XIII ) is:
             *
            ***
           *****
          *******
         *********
        ***********
       *************
      ***************
     *****************
    *******************
   *********************
  ***********************
*************************
The rotation for 90 degrees clockwise:
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
************
***********
**********
*********
********
*******
******
*****
****
***
**
*
```

```
Enter the size of your triangle (integer in [1, 50])
Type Q (or q) to quit the program: Q↲
Thank you, have a great day!
```

Demonstration and Submission

For the demonstration of Lab5, you need to pass the testing cases from the above sample outputs, and you also need to modify the value of the const **BASE** into 3, then recompile and test running your program again. It should match the following output:

```
Welcome to CS215 Roman Numeral Converter!
Decimal          Roman Numerals

      1      -->              I
      3      -->            III
      9      -->             IX
     27      -->           XXVII
     81      -->           LXXXI
    243      -->         CCXLIII
    729      -->         DCCXXIX
   2187      -->      MMCLXXXVII
   6561      -->      Error: NOT in the correct range!
  19683      -->      Error: NOT in the correct range!
  59049      -->      Error: NOT in the correct range!
 177147      -->      Error: NOT in the correct range!
 531441      -->      Error: NOT in the correct range!

Enter the size of your triangle (an integer in [1, 50])
Type Q (or q) to quit the program: q↲
Thank you, have a great day!
```

1. Each Lab assignment needs to demonstrate to your TA to be graded. You can demonstrate Lab5 during Lab5 class (with possible bonus 3 points) or no later than the end of Lab6 class (this is **the demonstration deadline** for Lab5).

If you finish Lab5 assignment during Lab5 class, you may demonstrate your program to your TA and answer your TA's questions, you can get up to 3 extra points for this lab assignment. (Note you can also demonstrate your program to your TA during Lab6 class. However, any demonstration later than the end of the Lab5 class cannot get bonus 3 points.)

If you need extra time, you can continue working on Lab5 assignment after the Lab class, and try to finish it before the next Lab class. Then demonstrate your Lab5 during Lab6 class.

If you do not demonstrate your code, even if you submit it in Canvas, you will receive a grade of 0!! *The TA may ask you to make some corrections. If so, make the corrections and demonstrate again…repeat until you have 100%!*

2. After the successful demonstration, submit the code in Canvas. Open the link to Course Canvas page (https://www.uky.edu/canvas), and log in to your account using your LinkBlue ID and password. Please submit your **source code in a .cpp file** through link "**`Lab 5`**".

Even if you successfully demonstrated it to the TA, if you do not submit in Canvas by the submission deadline, you will receive a grade of 0!

Grading (20 points + Bonus 3 points)

1. Attend the lab session or have a documented excused absence. (5 points)
2. Demonstrate your program to your TA and submit it in Canvas. (15 points)
   - Include comments for each function as specified in the lecture notes. (1 point)
   - Provide the correct definition of three functions. (each 3 points, total 9 points)
   - Provide the block of code in the main function correctly. (3 points)
   - Modify **`BASE = 3`**, re-compile the code and pass the testing case. (2 points)

Demonstrate your program to your TA and answer TA's questions during Lab class when the same Lab assignment is given. (Bonus 3 points)

Reference (solution from ChatGPT)

After you learn how to use arrays (as one type of data structure) to organize data items, this can be one of your solutions to Lab5. For now, it cannot be acceptable for this assignment.

```
#include <iostream>
#include <string>

std::string intToRoman(int num)
{
    std::string roman;

    // arrays to store Roman numeral symbols
    // and their corresponding values
    std::string romanSymbols[] = {"M", "CM", "D", "CD",
"C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
    int romanValues[] = {1000, 900, 500, 400, 100, 90, 50,
40, 10, 9, 5, 4, 1};

    for (int i = 0; i < 13; i++)
    {
        while (num >= romanValues[i])
        {
            num -= romanValues[i];
            roman += romanSymbols[i];
        }
    }
```

```
    return roman;
}
```

Together with a simple testing source file:

```
int main()
{
    int n;
    std::cout << "Enter an integer between 1 and 3999: ";
    std::cin >> n;

    if (n < 1 || n > 3999)
    {
        std::cout << "Invalid input! Please enter a number
between 1 and 3999." << std::endl;
    }
    else
    {
        std::string result = intToRoman(n);
        std::cout << "Roman Numeral representation: " <<
result << std::endl;
    }

    return 0;
}
```