Spring 2017 CMSC140 Programming Project 3: Formatting

Concepts tested in this project

- To work with Variables and Literals
- To learn and use different Data Type
- To learn and use Programming Style
- To learn and use Arithmetic Operators
- To work with cin command to get input
- To learn and use conditional statements
- To learn and use switch statements
- To learn and use relational operators

Project Description

This project consists of two parts.

Part 1: Ask the use to enter a number within the range of 1 and 5 and display the roman version of that number. If the number is not in the range then display an error message.

Part 2: The date June 10, 1960, is special when we write it in the following format, the month times day is equal to the year:

6/10/60

Ask the user to enter a month (in the numeric form), a day and a two-digit year and determine if the month times the day is equal to the year. If so, display the message "the date is a Magic Date!" otherwise display the message "the date is NOT a Magic Date.

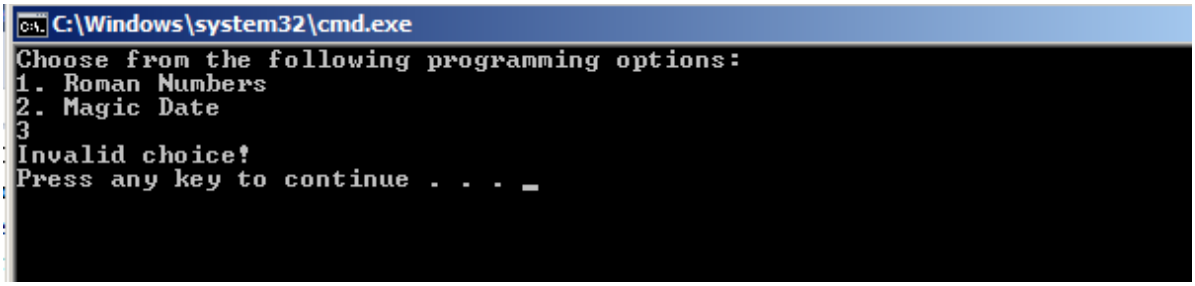
Perform input validation for month (ranges between 1 and 12), day (ranges between 1 and 31) and year (ranges between 10 and 99) and display error message for invalid inputs.

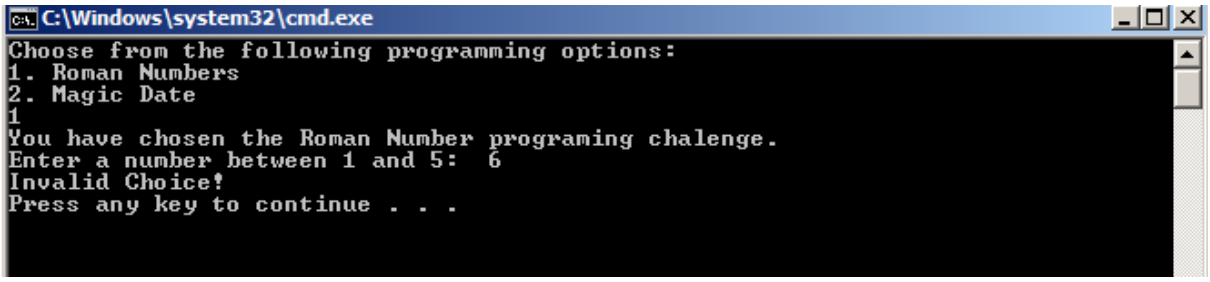
Write, compile and run a C++ program that displays the options of choosing either part1 or part 2 to the user and performs the specified task explained above depending on user's choice. **Refer to the screen shot of the sample output for more details**.

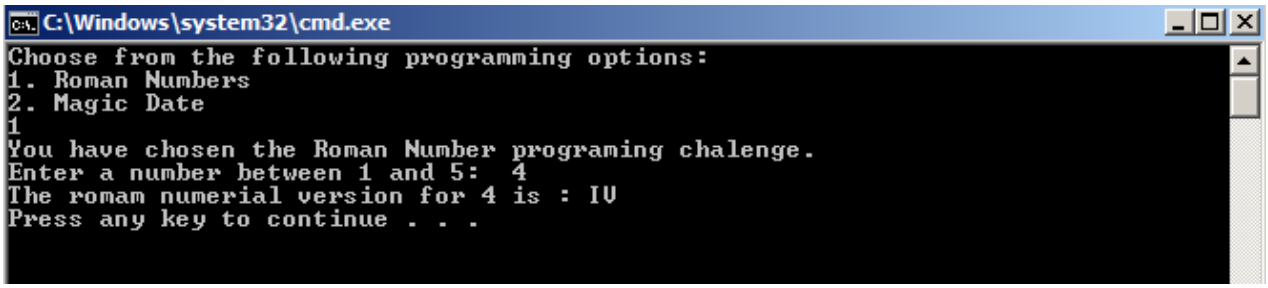
- Create a pseudo code of the program before you start coding to observe the flow of the program.
- Use a **switch statement** for determining and displaying the roman version of the number entered in **Part 1**.
- Perform input validation if the user enters and invalid choice and display an error message.

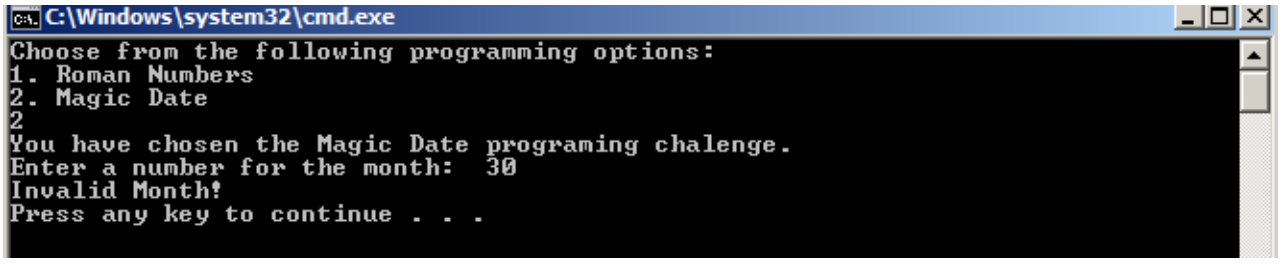
Spring 2017 CMSC140 Programming Project 3: Formatting

Following are sample runs of the program:

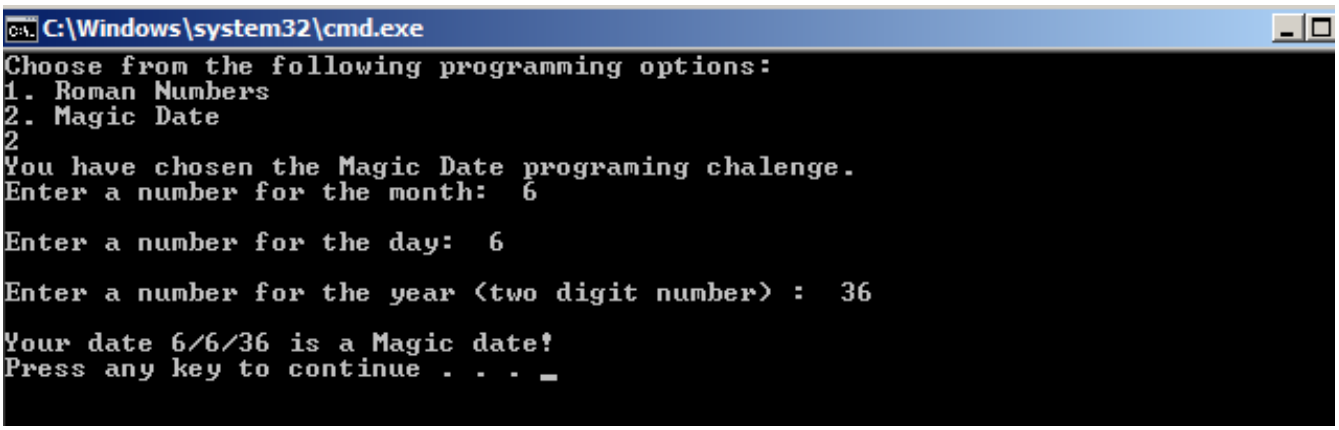








Spring 2017 CMSC140 Programming Project 3: Formatting



Project 3 Submission requirements:

Notes:

- Proper naming conventions: All constants, except 0 and 1, should be named. Constant names should be all upper-case, variable names should use "camel case" (i.e. start with lower case, with subsequent words starting with upper case: *hours**W**orked* for example) or underscores to separate words (i.e. items_ordered) (textbook, page 42)
- Variable and method names should be descriptive of the role of the variable or method. Single letter names should be avoided.
- Documentation: The documentation requirement for all programming projects is one block comment at the top of the program containing the course name and CRN, the project number, your name, project description, the due date and platform/compiler that you used to develop the project. If you use any code or specific algorithms that you did not create, a reference to its source should be made in the appropriate comment block. Additional comments should be provided as necessary to clarify the program.
- Indentation: It must be consistent throughout the program and must reflect the control structure.
- **Program Header**: You should include one block comment (header) at the top of each program containing the course name and CRN, Instructor's name, the project number, your name, the date and a short description of the project as follows:

```
/*
 * Class: CMSC140 CRN
 * Instructor:
*  Project [number]
 * Description: (Give a brief description for Project1)
 * Due Date:
  * I pledge that I have completed the programming assignment independently.
   I have not copied the code from a student or any source.
   I have not given my code to any student.
```

Spring 2017 CMSC140 Programming Project 3: Formatting

Print your Name here: _____
*/

Deliverables:

1. A Word document that includes:
   - Title Page with the following information
     - Project <#>, Due date (including year) , Your name, class, and section
   - Screenshots of the program
   - Source code of the program
   - Pseudocode or Flowchart for the program
2. Your source code (.cpp file). Your source code file should include a block comment (header) listed below.
3. The C++ files zipped and saved as LastNameFirstName_Project3_Moss.zip

   This .zip will not have any folders in it – only .cpp files.

   Note: This format is required to check for duplicate submissions using "MOSS" Plagiarism Detection Software.

Submit your completed assignment to **Blackboard** no later than the due date.

Grading Criteria for Project 3

This project will be graded using the following are components. **If program does not compile, project will get grade "0"**. Contact your instructor prior to the project submission due date, if you have compilation issues.

Attributes	**Value** (points)
Functionality (If project does not compile, project will get grade "0")	Total 100
Displays the user choices appropriately	15
The required information described in project description, are asked from the user	25
Input validation check on choosing part 1 or part 2	2
Input validation check on numbers entered for part 1	2
Input validation check on month, day and year for part 2	6
Use of switch statement for determining and displaying the roman number	10
Program executes correctly (produce expected output)	15
Meets all requirements	15
Overall Look-and-Feel	10
Total	Total 100 points

Spring 2017 CMSC140 Programming Project 3: Formatting

Project General Requirements (points will be deducted)

Attributes	Value(points)
Programming Style and proper naming convention: (see coding standards)	(-20 pts maximum)
Constants not all caps	-5
Curt or unclear variable names	-5
Long variable names should use camel case or underscores to separate words	-5
Comments and internal notes	
Sparse and inadequate comments.	-5
File header is not included	-5
Essentially no comments	-10
Indentation and white spaces should be a visual aid to understanding code structure	
Indenting is mostly okay, but sometimes inconsistent.	-5
No indenting, or very inconsistent indenting that is a barrier to understanding the code	-10
Lack of white space separating variables and operators. Lack of white spaces separating functions and major code blocks (later projects only)	-5
Test Plan	(-20 pts maximum)
Missing Entirely	-20
Cursory or inadequate testing	-10
Adequate overall, but missing a few crucial tests	-5
Missing Required Items (only if required for the project)	(-20 pts maximum)
Pseudocode, Flowcharts, or Hierarchy chart missing	-20
Screen shots cursory or incomplete	-5
Screen shots completely missing	-10
List of assumptions made (not applicable for Project 3)	-5
Highlights of your learning experience	-5
Awkward Code Internal Structure	(-10 pts maximum)
Hard-coding input values	-10
Poor structured programming: inappropriate loop choices, incorrect use of break statements to exit loops, and so on (not applicable for Project 3)	-5 each
Excessive reliance on global variables (e.g., using them to avoid pass by reference) (not applicable for Project 3)	-5
Processing array contents piecemeal rather than using loops (not applicable for Project 3)	-5
Other poor coding practices not mentioned	-5