Project #1 – When in Rome

|  |  |
| --- | --- |
| Course | INFO-1156 Object-Oriented Programming in C++ |
| Professor | Garth Santor, Lianne Wong, Janice Manning |
| Assigned | Monday, January 13th, 2020 |
| Due | Friday, January 31st, 2020 by 11:59 pm |
| Weight | 6% |

# Project Description

Write a C (**not C++**) console application that converts Roman numbers (e.g. XIV) to Arabic numbers (e.g. 24) or Arabic numbers to Roman numbers.

# Concepts

Roman numerals use a plus/minus system based on whether the smaller numeral precedes or follows the larger numeral.

## Base Values

|  |  |  |  |
| --- | --- | --- | --- |
| Numeral | Value | Numeral | Value |
| I | 1 | V | 5 |
| X | 10 | L | 50 |
| C | 100 | D | 500 |
| M | 1000 |  |  |

## Combining rule

Numeral values are added together, unless the numeral is followed by a greater numeral. In this case, the smaller numeral value is subtracted from the following value.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| Roman | Arabic | Roman | Arabic |
| III | 3 | VI | 6 |
| IV | 4 | XCII | 92 |

## Fractions

The romans represented fractions using two symbols, ‘S’ for *semis* – a half, and ‘∙’ for an uncia – a twelfth.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| Roman | Arabic | Roman | Arabic |
| IIIS |  | VI∙∙ |  |
| IVS∙∙∙ |  | XCIIS∙∙∙∙ |  |

We’ll have to use periods for the dots, since a dot is not easily typed on a standard keyboard.

### Vinculum

The apostrophus system is very limited in the possible number that can represented. The vinculum system writes a bar above the numerals indicate that the numerals beneath should be multiplied by 1,000. Thus, is 4,000.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| Roman | Arabic | Roman | Arabic |
|  | 4,000 |  | 4,020 |

### Apostrophus (Balanced)

The earlier method for representing large numbers is the *apostrophus*. In this system, the number five hundred ‘D’ is represented by an ‘I’ and a backward c – ‘Ↄ’ (D = IↃ). One thousand (1000) was written as CIↃ. Adding C’s to either side multiplies the number by 10.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| Roman | Arabic | Roman | Arabic |
| CIↃ | 1,000 | CCCIↃↃↃ | 100,000 |
| CCIↃↃ | 10,000 | CCCCIↃↃↃↃ | 1,000,000 |
| CCIↃↃLII | 10,052 | CCCCIↃↃↃↃI | 1,000,001 |

If there are additional backwards c’s, we add a multi of five.

### Examples (Unbalanced)

|  |  |  |  |
| --- | --- | --- | --- |
| Roman | Arabic | Roman | Arabic |
| IↃ | 500 | IↃↃↃ | 50,000 |
| IↃↃ | 5,000 | IↃↃↃↃ | 500,000 |
| IↃↃDLV | 5,555 | IↃↃↃↃV | 500,005 |

These can be combined…

### Examples (Unbalanced)

|  |  |  |  |
| --- | --- | --- | --- |
| Roman | Arabic | Roman | Arabic |
| CIↃↃ | 1,500 | CCIↃↃↃↃ | 15,000 |
| CCIↃↃↃ | 10,500 | CCCIↃↃↃↃↃↃ | 150,000 |
| CCIↃↃↃXXI | 10,521 | CCCIↃↃↃↃↃↃIV | 150,004 |

# Requirements

Write a C program (no C++, Visual Studio 2019 solution) that reads either a Roman number or an Arabic number and then converts the number to the other representation (automatically detects the direction of the conversion).

The *fraction* notation will use periods instead of dots.

The *apostrophus* notation will use a right parenthesis ‘)’ for the backward c ‘Ↄ’ as the backward c is not part of the ASCII character set.

The *vinculum* notation will use a dash or underscore following the numerals that it modifies. (e.g. XX- and XX\_ are 20,000, XX-XX is 20,020, and IV\_XX is 4,020).

The program should repeat the conversions until the user types ‘quit’ (less marks for a non-repeating program).

The output should print only as many decimals places as necessary.

The program input should be case insensitive.

## References

<https://en.wikipedia.org/wiki/Roman_numerals>

# Grading Criteria

|  |  |  |
| --- | --- | --- |
| **Functional Requirements** |  |  |
| Program converts basic Roman to Arabic:   * converts simple expressions (e.g. XX or VI) * converts prefix expressions (e.g. IV or XC) | 10%  10% |  |
| Program converts Arabic to basic Roman:   * converts simple expressions (e.g. XX or VI) * converts prefix expressions (e.g. IV or XC) | 10%  10% |  |
| Program converts fractions:   * Roman to Arabic (e.g. S... becomes 0.75) * Arabic to Roman (e.g. 0.75 becomes S…) | 5%  5% |  |
| * No rounding errors | 5% |  |
| Program converts vinculum numerals:   * Vinculum Roman to Arabic (e.g. XVI- or XVI\_ becomes 16,000) * Arabic to Vinculum Roman (e.g. 37,015 becomes XXXVII\_XV) | 5%  10% |  |
| Automatic detection of conversion direction. | 10% |  |
| Program repeats until ‘quit’ is entered. | 5% |  |
| Program doesn’t print unnecessary trailing zeros. | 5% |  |
| Program converts balanced *apostrophus* numerals:   * to Arabic (e.g. CIↃ becomes 1,000, CCCIↃↃↃ becomes 100,000) | 5% |  |
| Program converts unbalanced *apostrophus* numerals:   * to Arabic (e.g. IↃↃ becomes 5,000, IↃↃↃↃ becomes 500,000) | 3% |  |
| Program converts combined unbalanced *apostrophus* numerals:   * to Arabic (e.g. CIↃↃ becomes 1,500, CCIↃↃↃↃ becomes 15,000) | 2% |  |
| **Non-functional requirements** |  |  |
| Executable program is not named ‘**roman.exe**’ | -10% |  |
| Penalties from *C & C++ Grading Guide* ***v2.1.0*** | various |  |
| Late submission   * One to five days late * More than five days late | -10%/day  -100% |  |
| **Total** | **100%** | **100%** |

# Submission Requirements

1. Submit **entire Visual Studio project directory** to Fanshawe Online
   1. Delete ***all*** debug and release directories.[[1]](#endnote-1)
   2. Submit in a .ZIP, .7z archive file.

1. Alternatively, you can ‘clean’ your project for submission by downloading ‘vsclean’ a Visual Studio Solution Cleaner from <https://www.gats.ca/software/vsclean/> . [↑](#endnote-ref-1)