Assignment 3 – Strings and Classes

1. Roman:

Background:

The following section introduces Roman numerals, conversions between Roman numerals and Base 10, and addition in Roman numerals.

Introduction to Roman Numerals:

The decimal numbering system is a positional system created with a base 10 as you've seen in your math class. Hence the 1 in 123 compared to the 1 in 15 mean very different things. As a position goes we could re-write decimal to be 100 + 20 + 3 and 10 + 5, where each column is associated with 10^p. If we change the position we change the meaning of the number.

Roman Numerals however is not a positional system. Each symbol takes on its numerical value regardless of where it appears, there is no 0, and there are a few subtractive rules for short hand that will be demonstrated below.

The general symbols are:

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Table 1.0: Roman Numeral Basic Units and Base 10 equivalent

For example "DCCLXVII" is equivalent to 767 as demonstrated by: 500 + 100 + 100 + 50 + 10 + 5 + 1 + 1 = 767 (base 10) using Table 1.0 above. Because it is position based CCLXVIID would be the same 767 in base 10.

Of course, the Romans would not have been able to think in terms of our numeric values (i.e. 767). They could only manipulate the symbols directly. As an example, they would not have known that **VV** is an **X** because 5 + 5 = 10. They would have needed to memorize and use a grouping equivalence table:

IIIII	is equivalent to	**V**
VV	is equivalent to	**X**
XXXXX	is equivalent to	**L**
LL	is equivalent to	**C**
CCCCC	is equivalent to	**D**
DD	is equivalent to	**M**

Table 2.0: Roman Numeral Groupings

Such equivalence would have reduced the number of symbols in a number. As an example, 600 could be written as **CCCCCC** (or 100+100+100+100+100+100).This would have been "simplified" by the equivalence rules to the properly written **DC** (or 500+100). For that matter, 600 could be expressed as 12 **L**'s or 60 **X**'s or 120 **V**'s or 600 **I**'s or some combination as long as the sum of all the characters totaled 600. In all cases, by applying the equivalence rules to shorten the expression by replacing several symbols with one, we would get **DC**.

Simply applying the rules so far could lead to a Roman value such as **VIIII** (or 9). To write this more compactly, we use the convention that we can "subtract" a symbol representing a 1, 10, or 100 from the next **two** higher symbols, respectively, by writing the smaller to the left. Therefore the **only** subtractive forms are:

Write	Instead of	Base 10 Value
IV	**IIII**	4
IX	**VIIII**	9
XL	**XXXX**	40
XC	**LXXXX**	90
CD	**CCCC**	400
CM	**DCCCC**	900

Table 3.0: Roman Numeral Subtractive

Therefore, as an example: "**MCMXCIV"** would be: **1000 + 900 + 90 + 4 = 1994**
The 900, of course, comes from the pair of symbols: **CM**.
The Roman value is equivalent to: **MDCCCCLXXXXIIII**. While this is less compact, we shall see that this equivalent form without any subtractives can be useful.

Conversion – From Roman to Base 10:
Simply add up the values of the Roman symbols. Of course, if a subtractive appears with a **I, X,** or **C** to the left of a "larger" symbol, we need to substitute the pair for the correct numeric value. The conversion example of 1994 above is an example of going from Roman to Base 10. The general process:

- Get the Roman Number:
  MCMXCIV

- Remove any subtractive terms that might exist (expand and replace)
  MCMXCIV = MCMXCIIII (order of conversion doesn't matter)
  MCMXCIIII = MCMLXXXXIIII (Use Table 3.0)
  MCMLXXXXIIII = MDCCCCLXXXXIIII
  Thus, **MCMXCIV = MDCCCCLXXXXIIII** (with subtractive terms removed)

- Convert each value to its base 10 equivalent adding to your base 10 variable
  M D C C C C L X X X X I I I I
  1000 + 500 + (100+100+100+100) + 50 + (10+10+10+10) + (1+1+1+1)

  Thus,**MDCCCCLXXXXIIII = 1994** (in Base 10, which is implied)

Conversion – From Base 10 to Roman:
To convert from Base 10 to a Roman numeral start by removing the largest Roman values first and subtracting the removed value until we have converted the entire value. We keep trying to remove a given Roman value until we cannot, then we try the next smaller one. As an example we will convert 2349 Base 10 to Roman. Pay particular attention to the result logic.

The table below indicates the steps a computer would move through. A human would normally take some short cuts by lumping. As an example, we readily see that there are two "thousands" in the original value and would immediately write down: **MM**. There are three "hundreds" so we get **MMCCC**, etc. until we get the entire number converted.

Base 10 value	Symbol to Try	Result Ok?	Value Remaining	Roman Value
2349	**M** or 1000	yes	1349	**M**
1349	**M** or 1000	yes	349	**MM**
349	**M** or 1000	no	349	**MM**
349	**D** or 500	no	349	**MM**
349	**C** or 100	yes	249	**MMC**
249	**C** or 100	yes	149	**MMCC**
149	**C** or 100	yes	49	**MMCCC**
49	**C** or 100	no	49	**MMCCC**
49	**L** or 50	no	49	**MMCCC**
49	**X** or 10	yes	39	**MMCCCX**
39	**X** or 10	yes	29	**MMCCCXX**
29	**X** or 10	yes	19	**MMCCCXXX**
19	**X** or 10	yes	9	**MMCCCXXXX**
9	**X** or 10	no	9	**MMCCCXXXX**
9	**V** or 5	yes	4	**MMCCCXXXXV**
4	**V** or 5	no	4	**MMCCCXXXXV**
4	**I** or 1	yes	3	**MMCCCXXXXVI**
3	**I** or 1	yes	2	**MMCCCXXXXVII**
2	**I** or 1	yes	1	**MMCCCXXXXVIII**
1	**I** or 1	yes	0	**MMCCCXXXXVIIII**

There is one more step we need to do. We need to substitute for any subtractive values. To do this we proceed right-to-left and look for **four** of the same symbol. Since there are four **I**'s we check the next symbol on the left and substitute **IX** for the **VIIII**. (Note, if the Roman value were something like, **MDIIII**, we could not utilize the **D** so we would end up with **MDIV**.) After making this substitution we have: **MMCCCXXXXIX**.

Thus the process would be:

- Get the Base 10 Number
  2349

- Perform the operation from the table above to find the Roman Numeral
  2349 = MMCCCXXXXVIIII

- Process the Roman Numeral to implement the subtractive
  MMCCCXXXXVIIII = MMCCCXXXXIX (Use Table 3.0)
  MMCCCXXXXIX = MMCCCXLIX (4 X with a preceding C, becomes XL)

  Thus, **MMCCCXXXXVIIII = MMCCCXLIX** (Final answer, reduced form)

Hint: *The order of processing will matter. If you find XXXX it makes a difference if it is XXXX (XL) or LXXXX (XC). If you have LXXXX (90) and you find and replace the XXXX (40) first you will be left with LXL rather than the simplified XC.*

Operation - Addition:

If you were asked to **add** two Roman valves such as **CXXII + LXI**, you would probably convert these two values to base 10 numbers, add them (122 + 61 = 183), and finally convert back to Roman numerals: **CLXXXIII**. This is because it is relatively easy to convert and we know how to add ordinary decimal numbers.

The Romans could not do this! They needed a method of manipulating the Roman symbols directly to achieve the addition!

It turns out that an algorithm for adding Roman numbers directly is actually quite easy. This was fortunate for the Roman engineers and accountants.

The algorithm has just five steps:

1. Substitute for any subtractives in both values; that is; "uncompact" the Roman numerals
2. Put the two values together—concatenate them
3. Sort the symbols in order from left-to-right with the "largest" symbols on the left
4. Starting with the right end, combine groups of the same symbols that can make a "larger" one and substitute the single larger one
5. Compact the result by substituting subtractives where possible

As an example, perform **CCCLXIX + DCCCXLV**.

1.	Substitute for any subtractives to obtain:	**CCCLXVIIII + DCCCXXXXV**
2.	Concatenate to obtain:	**CCCLXVIIIIDCCCXXXXV**
3.	Sort to obtain:	**DCCCCCCLXXXXXVVIIII**
4.	Combine groups to obtain:	**DCCCCCCLXXXXXXIIII** **DCCCCCCLLXIIII** **DCCCCCCCXIIII** **DDCCXIIII** **MCCXIIII**
5.	Substitute any subtractives to obtain:	**MCCXIV**

You can verify that this is correct by converting the values to regular notation: 369 + 845 = 1214.

Requirements:

Write an application that conducts the following:

- Provides a menu that is looped for user selection ensuring out of bounds exceptions are caught
- The menu will allow the user to select:
  - 1. Roman numeral to Base 10
  - 2. Base 10 to Roman numeral
  - 3. Add 2 Roman numerals
  - 4. Exit
- The "Roman numeral to Base 10" will:
  - Conduct Roman numeral to base 10 conversions as per the above background, operation to be conducted in a function called "RomantoBase10"
  - The entry should be constrained to an equivalent maximum of 9,999 and a minimum entry of 1. Display both the Roman numeral and base 10 solution to the screen to demonstrate to the user the process that was under taken.

    - You have two logical approaches available:
        - If no subtractive logic (ie. CCCLXVIIII = 369)
        - If subtractive logic (ie. MCMXCIV = 1994)
- The "Base 10 to Roman numeral" will:
    - Conduct Base 10 to Roman numeral conversions as per the above background, operation to be conducted in a function called "Base10ToRoman"
    - Allow the user to enter a value to convert or the application to provide a random value. Either entry should be constrained to a maximum of 9,999 to convert with a minimum entry of 1. All entries should be limited to positive integer values only. Display both the base 10 and Roman numeral solution to the screen to demonstrate to the user the process that was under taken.
    - You have two logical approaches available:
        - If no subtractive logic (ie. 845 = DCCCXXXXV)
        - If subtractive logic (ie. 2349 = MMCCCXLIX)
- The "Add 2 Roman numerals " will:
    - Add 2 Roman numerals together using a function called "RomanAddition"
    - The same data constraints exist as per the above functions
    - You have two logical approaches available:
        - Without subtractive logic
        - With subtractive logic
- You will also need two additional functions called "RemoveSubtractives" and "AddSubtractives".
    - These functions will need to be called from multiple locations in the application to prepare the data for manipulation during the above conversions
    - Use the background above for context with the working examples
- The "Exit" will:
    - Terminate the application

Hint: *You can use arrays/for loops to help with the subtractive. While a structure is not required feel free to use any tool at your disposal through class.*
Hint: *There are many ways to do Roman Addition in this example. You can use any method possible and provided the inputs work (with or without subtractive) the marks will be awarded respectively.*

Grading (Total 40 points):

- 3 points for code style (i.e. does it follow good coding practises, naming conventions etc.)
- 3 points for clarity of code (comments, white space, indents, etc).
- 4 points for the menu, looping, acceptable input, and exit capabilities
- 10 points for the "RomantoBase10" requirements with subtractive, 5 points if done without
- 10 points for the "Base10ToRoman" requirements with subtractive, 5 points if done without
- 10 points for the "RomanAddition" requirements with subtractive, 5 points if done without

Sample: //sample user input provided below in red (The __ are not required, here for clarity)

```
Gratissimum Jason                          //Welcome message displayed once

Please select one of the following:        //menu that is repeated

1. Roman to Base 10
2. Base 10 to Roman
3. Add 2 Roman numbers
4. Exit

Indicate your selection: 1                 //Ensure the input is the right 1-4 range every time

_____
Input the Roman number: MCMXCIV            //RomanToBase10 function call
In Base 10 this is: 1994                   //no need to show work (but you can), just the solution for all
_____


Please select one of the following:        //menu is repeated to allow user to input again

1. Roman to Base 10
2. Base 10 to Roman
3. Add 2 Roman numbers
4. Exit

Indicate your selection: 2

___
Input the base 10 number (< 9999 and > 0): 2349        //Base10ToRoman function call
The Roman number is: MMCCCXLIX             //NB. The range controls, you should check the user input for conformance

___
Please select one of the following:        //menu is repeated to allow user to input again

1. Roman to Base 10
2. Base 10 to Roman
3. Add 2 Roman numbers
4. Exit

Indicate your selection: 3

_____
Input the first Roman number: CCCLXIX      //Roman Additional Function call
Input the second Roman number: DCCCXLV     //We will only ever be adding 2 Roman numbers in our software
The sum of CCCLXIX + DCCCXLV = MCCXIV      //this is just a cout statement with your various strings outputted


____
Please select one of the following:        //menu is repeated to allow user to input again

1. Roman to Base 10
2. Base 10 to Roman
3. Add 2 Roman numbers
4. Exit

Indicate your selection: 4

Gratias tibi!!                             //good bye message, only shown once

Press Any Key to Continue ....
```

Save the source for number 1 as “Assn3-Roman-YourName”

Reference: http://turner.faculty.swau.edu/mathematics/materialslibrary/roman/

2.Encode:

Requirements:

Write an application that conducts the following:

- Provides a menu that is looped for user selection ensuring out of bounds exceptions are caught
- The menu will allow the user to select:
  - 1. Enter the string data
  - 2. Encrypt the string
  - 3. Decrypt the string
  - 4. Exit
- The "Enter the string data" will:
  - Allow the user to enter string based data. You can use the string class.
  - If the user selects this option again over-write the original string.
  - For the purposes of this assignment store the string in a class which will be used to contain the original, the encrypted string, and a prepared string with spaces removed (per below) as required. The class should also contain at a minimum a display function to print encrypted data and original data to the screen. Other functions placement will be your choice.
- The "Encrypt the string" will:
  - Ask the user what cipher they would like to use for encryption:
    - 1. XOR Cypher
      - Apply the XOR Cypher to encrypt the string provided by option 1
      - If there is no string stored in Option 1 (main menu) do not run any encryption but inform the user there is no data to encrypt
      - Allow the user to provide a letter to be used as the key. We will only take a single character as a key although a list could be used, etc.
      - Skip any spaces in the string, thus formatting will be maintained. See sample below.
    - 2. Caesar Cypher (Shift Cypher)
      - Apply the Caesar Cypher to encrypt the string provided by option 1
      - If there is no string stored in Option 1 (main menu) do not run any encryption but inform the user there is no data to encrypt
      - Allow the user to provide a key to shift with. You only need a single key for this cypher although a key list could also be used.
      - The key must be limited between 0 and 25 (to roll-over the alphabet)
      - Skip any spaces in the string, thus formatting will be maintained. For example: "Hi this is test" could be "Ij uijt jt uftu" with the maintained spacing.
    - 3. Vigenère Cypher
      - Apply the Vigenère Cypher to encrypt the string provided by option 1
      - If there is no string stored in Option 1 (main menu) do not run any encryption but inform the user there is no data to encrypt. For this case remove all spaces that may exist in the user provided string and ensure all characters are lower case. Do not over-write the original data string but rather use a copy that you can manipulate. I.e. "Hi this is a test" will be "hithisisatest"
      - Allow the user to provide a string as the key word. Verify, that this key word does not contain any spaces, only consists of characters a to z, force all characters to be lower case, and that the keyword is at least 3 characters long (if it is less ask the user for another key word). The look up table (tabula recta) is included on the last page.
      - On the screen demonstrate the original string, the prepared string of original text over-layed with the key, then the encrypted message below.
  - Following each encryption the original message and the encrypted message should be displayed together.

- The string class and math library can be used (if required) however no pre-created algorithms should be used to solve this question.

- The "Decrypt the string" will:
  - Ask the user what cipher they would like to use for decryption:
    - 1. XOR Cypher
      - Apply the XOR Cypher to decrypt the string containing encrypted information. If there is no encrypted data in the string inform the user to encrypt data first.
      - Allow the user to provide a single character key
      - Skip any spaces in the string, thus formatting will be maintained.
      - Use the provided key to decrypt whatever is in the encrypted string using the XOR Cypher. If it is a match with the XOR Cypher the original message will be displayed. In either capacity complete 1 full iteration of the decryption and display the results to the screen.
    - 2. Caesar Cypher (Shift Cypher)
      - Apply the Caesar Cypher to decrypt the string containing encrypted information. If there is no encrypted data in the string inform the user to encrypt data first.
      - Allow the user to provide a key to shift with that is limited between 0 and 25.
      - Skip any spaces in the string, thus formatting will be maintained.
      - Use the provided key to decrypt whatever is in the encrypted string using the Caesar Cypher. If it is a match with the Caesar Cypher the original message will be displayed. In either capacity complete 1 full iteration of the decryption and display the results to the screen.
    - 3. Vigenère Cypher
      - Apply the Vigenère Cypher to decrypt the string containing encrypted information. If there is no encrypted data in the string inform the user to encrypt data first. If there are any spaces in the data string do not use this cypher to decrypt but rather inform the user that they should attempt either the XOR or Caesar Cyphers.
      - Allow the user to provide a string as the key word. Verify, that this key word does not contain any spaces, only consists of characters a to z, force all characters to be lower case, and that the keyword is at least 3 characters long (if it is less ask the user for another key word)
      - On the screen demonstrate the original encrypted string, the prepared string of original text over-layed with the key, then the decrypted message below.

- The "Exit" will:
  - Terminate the application.

Grading (Total 50 points):

- 3 points for code style (i.e. does it follow good coding practises, naming conventions etc.)
- 3 points for clarity of code (comments, white space, indents, etc).
- 4 points for the menu, looping, acceptable input, and exit capabilities
- 5 points for the class requirements and use
- 10 points for the XOR Cypher
- 10 points for the Caesar Cypher
- 15 points for the Vigenère Cypher

Sample: //sample user input provided below in red (The __ are not required, here for clarity)

Welcome to encoder!

1. Enter the string data
2. Encrypt the string
3. Decrypt the string
4. Exit

Please make your selection: 2

No original data was found. Please ensure that you enter the string prior to encrypting or decrypting.

---

1. Enter the string data
2. Encrypt the string
3. Decrypt the string
4. Exit

Please make your selection: 3

No original data was found. Please ensure that you enter the string prior to encrypting or decrypting.

---

1. Enter the string data
2. Encrypt the string
3. Decrypt the string
4. Exit

Please make your selection: 1

Please enter your original message: Hi this is a test

---

1. Enter the string data
2. Encrypt the string
3. Decrypt the string
4. Exit

Please make your selection: 3

Original data has been provided but it has not been encrypted. Please encrypt prior to decrypting.

---

1. Enter the string data
2. Encrypt the string
3. Decrypt the string
4. Exit

Please make your selection: 2

What Cypher would you like to use?
1. XOR Cypher
2. Caesar Cypher
3. Vigenère Cypher

Please make your selection: 3 //sample selected for more complex option

Please enter the key word: MYk

Sorry this key is invalid. Please try another keyword that consists of 3 or more characters.

Please enter the key word: MYkey

Original Text: Hi this is a test
Keyword: mykeymykeymyk
Prepared Text: hithisisatest
Encrypted Text: tgdlgugcerqqd

---

1. Enter the string data
2. Encrypt the string
3. Decrypt the string
4. Exit

Please make your selection: 3

What Cypher would you like to use?
1. XOR Cypher
2. Caesar Cypher
3. Vigenère Cypher

Please make your selection: 3

Key word: mykeymykeymyk
Encrypted Text: tgdlgugcerqqd
Original Text: hithisisatest

Save the source for number 2 as “Assn3-Encode-YourName”

Appendix: Vigenère Cypher tabula recta

The Vigenère Cipher is a polyalphabetic substitution cipher which was conceptually developed in 1553.

The below table is used for the encryption:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	**L**	M	N	O	P	Q	R	S	**T**	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	**G**	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	**D**	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

To encrypt a message:

- repeat the keyword above the input string as demonstrated below
- Take the letter of the prepared text (i.e. h) and find it in the left hand column of the table above.
- Move along the corresponding row until you reach the column that has “m” at the top based on the keyword.
- Where these intercept is the encrypted letter

The first few letters have been colour coded and bolded to demonstrate where they are from.

Keyword:	m	y	k	e	y	m	y	k	e	y	m	y	k
Prepared Text:	h	i	t	h	i	s	i	s	a	t	e	s	t
Encrypted Text:	t	g	d	l	g	u	g	c	e	r	q	q	d

To decrypt a message the opposite approach is taken:

- Use the key letter (i.e. m) and find t in the column
- Go to the left to find the corresponding letter in the first column (ie. h)