CS 202 - Spring 2018
Dr. Williams
Assignment 3
Grade formula: SCE * (CS + 2 * VS + 3 * PD + 1)
Maximum raw score: 4 * (4 + 2 * 4 + 3 * 4 + 1) = 100
Maximum scaled score: 80

You are to write a program utilizing a class called `roman` which will be used to store a Roman numeral and convert it to Arabic numerals. If you are not familiar with Roman numerals, see https://www.periodni.com/roman_numerals_converter.html

The class should contain **only** the following `private` member variables:

- `roman_numeral`: a `string` that contains a Roman numeral or the null string
- `valid`: a `bool` that represents if the Roman numeral stored within is valid

The class should contain **only** the following `public` member functions:

- The default constructor which sets `valid` to `false`
  - There is no need to set the value for `roman_numeral` because the default `string` constructor creates an empty string
- `void set_roman(string)`: if the `string` parameter contains only characters used in Roman numerals (capital M, D, C, L, X, V, and I) it should set `roman_numeral` equal to the `string` value and set `valid` equal to `true`
- `string get_roman() const`: returns `roman_numeral`
- `int get_arabic()`: if `valid` is `true`, does the conversion of `roman_numeral` to Arabic numerals and returns it, otherwise returns 0

A data file will contain an unknown number of lines to be read and processed as described below. With respect to the file:

- You must get the name of the file via command line argument
- If no filename is provided then an error should display
- If a filename is provided but it does not exist then a different error should display
- If a filename is provided and does exist then it is guaranteed to contain at least one line
- Each line in the file will contain one potential Roman numeral to be processed
- The file will contain no blank lines but will end with a newline
- If a line in the file contains only characters used in Roman numerals then it is guaranteed to be a Roman numeral conforming to standards (e.g. nothing like VVVVM or MVVVV)

After your argument and file verification your main body should contain this **exact** code snippet:

```
string line;
while (getline(infile, line))
{
     roman r;

     r.set_roman(line);

     cout << "Line read: " << line << endl;
     cout << "Roman numeral stored: " << r.get_roman() << endl;
     cout << "Arabic numeral calculated: " << r.get_arabic() << endl;
}
```

Additional details:

- Utilize information hiding techniques using 3 files:
    - Class definition / interface file (called *loginname*.roman.h)
    - Class implementation file (called *loginname*.roman.cpp)
    - Main program file (called *loginname.main.cpp*)
- In addition to regular comments for non-trivial code and a comment for each function, each of the above files above should contain your name and course information as well as details about what each file does and is used for
- Create a makefile (called `makefile`) that does the following:
    - Compiles your class to object code (*loginname*.roman.o by default)
    - Links your class and compiles your main program to `loginname.out` (use the `-o` option)
    - Both compilation lines must use the compiler flags that we will use for every program: `-std=c++11 -Wall -Wextra -Wpedantic -Werror`
    - Place a comment at the top of your makefile containing your name
- `tar` your files together as follows:
    - `tar -cvf a3.tar` *loginname*`.roman.h` *loginname*`.roman.cpp` *loginname*`.main.cpp makefile`
- Verify your `tar` file contains exactly 4 files above and they are the correct ones:
    - `tar -tf a3.tar`
- Submit your `tar` file to `williams` on bobby with assignment code `03` by the deadline
- Printouts should contain the following in this order:
    - Cover sheet
    - Makefile
    - Class definition file
    - Class implementation file
    - Main program file
- If you end up comparing hard-coded values to the result of the `string.length` function, please note that `string.length` returns an `unsigned int` -- compiler errors may result if you try to compare it to a normal `int`
- A PD deduction of 1 will occur if you do not implement the “subtraction” part of the coversion (e.g. LXVI is simply 50 + 10 + 5 + 1, but LXIV is 50 + 10 + 5 - 1)

See the example compilation, execution, test, and submission of a program on the following page with my input in bold.

```
[williams@bobby 03]$ ls
data  makefile  williams.main.cpp  williams.roman.cpp  williams.roman.h
[williams@bobby 03]$ cat data
MMLX
VIII
opapjfjas
XV
XXIX
[williams@bobby 03]$ make
g++ -std=c++11 -Wall -Werror -Wpedantic -Wextra -c williams.roman.cpp
g++ -std=c++11 -Wall -Werror -Wpedantic -Wextra -o williams.out williams.main.cpp
williams.roman.o
[williams@bobby 03]$ ls
data  makefile  williams.main.cpp  williams.out  williams.roman.cpp  williams.roman.h
williams.roman.o
[williams@bobby 03]$ ./williams.out
Error, filename not provided!
[williams@bobby 03]$ ./williams.out bbbbbbb
Error, file could not be opened!
[williams@bobby 03]$ ./williams.out data
Line read: MMLX
Roman numeral stored: MMLX
Arabic numeral calculated: 2060
Line read: VIII
Roman numeral stored: VIII
Arabic numeral calculated: 8
Line read: opapjfjas
Roman numeral stored:
Arabic numeral calculated: 0
Line read: XV
Roman numeral stored: XV
Arabic numeral calculated: 15
Line read: XXIX
Roman numeral stored: XXIX
Arabic numeral calculated: 29
[williams@bobby 03]$ tar -cvf a3.tar williams.roman.h williams.roman.cpp
williams.main.cpp makefile
williams.roman.h
williams.roman.cpp
williams.main.cpp
makefile
[williams@bobby 03]$ tar -tf a3.tar
williams.roman.h
williams.roman.cpp
williams.main.cpp
makefile
[williams@bobby 03]$ submit a3.tar 03 williams
dos2unix: Binary symbol found at line 1
dos2unix: Skipping binary file /opt/submit/tmp_williams/williams.03.20180131083122
a3.tar Successfully Submitted!
```