CS 7B - Spring 2016 - Assignment 1. 2/4/16

In *Programming Practices and Principles*, Chapter 10

1. *Programming Principles and Practices* Chapter 10 exercise 6:
   Define a Roman_int class for holding Roman numerals (as ints) with a << and >>. Provide Roman_int with an as_int() member that returns the int value, so that if r is a Roman_int, we can write
   `cout << "Roman" << r << " equals " << r.as_int() << '\n';`
   It may help to look at RomanNumsCase paper...at least for a good definition of what a Roman numeral is. The recursive `Scheme` code is worth a look at...`Scheme` is a great language, built on recursion from the ground up. But we write C++.
   You're not required to validate the Roman numeral, though it is a good feature for such a class.
   I found it easier to read from least significant to most significant. Plus, I used a function

```
int as_int(char c) {
        switch (toupper(c)) {
            case 'I':  return 1;
            case 'V':  return 5;
            case 'X':  return 10;
            case 'L':  return 50;
            case 'C':  return 100;
            case 'D':  return 500;
            case 'M':  return 1000;
        }
        throw("Invalid character");
    }
```

   I then overloaded this function like so:

```
int as_int() {
    //initialize variables
    //use a for-loop to read from the end of the RN string backwards
    //calling as_int(RN[i])) to get the value
    //and adding it to the running total if it greater than the previous value
    //or subtracting is it's less
    // For example XLIV would read the V and add 5. Then, since I<V, 1 is
    // subtracted. Now 50 is added for the L, but since X<L, 10 is subtracted
    // and we have XLIV = 5-1+50-10 = 44.
}
```

   Include as a comment at the end of your code the results of a trial run or two like that shown above. Send your `.cpp` file to my email address with the format `<your initials>_Roman.cpp` by Tuesday, February 16 at 10:45 am.