CECS 282 - Homework 8

Complete these problems on a separate sheet of paper. Due March 24.

1. Reading from *C++ How to Program*:
   (a) Chapter 9.6, 9.7
   (b) Chapter 17.1, 17.2, 17.5
   (c) *Skim* Chapter 10.10, 10.10.1, 10.12

2. There is a subtle memory leak in the following block of code. Describe under what circumstances the block will leak (heap) memory, and how to resolve the issue.

```
try {
   cout << "Enter a number less than 10" << endl;
   int *x = new int;
   cin >> *x;
   if (*x >= 10)
      throw std::out_of_range("listen to the instructions dummy");
   delete x;
}
catch (std::out_of_range &ex) {
   cout << "You didn't listen!"  << endl;
}
```

3. In your own words, describe how you would decide whether to create a new variable on the stack or on the heap. (Look it up.)

4. Show how to implement `operator std::string()` as a member operator of your Rational class. `Rational::operator std::string()` should return a `std::string` representing the Rational object using the same rules as the `ToString()` method; if implemented correctly, you could *change* `ToString()` to be a single statement, `return (std::string)*this;`, which would cast the Rational object to a string using your conversion operator.

5. Suppose you have a class `Point`, representing a point in the 2D coordinate plane, with public member variables `int x` and `int y`, as such:

```
class Point {
public:
   int x;
   int y;
};
```

   Show how to implement an `operator-`, taking two Point objects as parameters, which returns the **Euclidian distance** between the two points. Decide an appropriate return type for the operator.

6. Answer True or False to the following questions:
   (a) You **must** declare a destructor for *every* class you make.
   (b) If you do not define a copy constructor, the compiler will automatically define one for you.
   (c) A compiler-defined copy constructor knows how to perform deep-copies of member variables allocated on the heap.
   (d) Managing memory on the heap is **super easy** and there is **never** a reason to program in any language that does garbage collection.