CECS 282 - Lab 6

Reading

Reading from *C++ How to Program*:

1. Chapter **10.9**
2. Chapter 10.10, 10.10.1, 10.12, 10.15

Assignment

1. Consider the following short main function, using your `BoardPosition` class from Homework 4:

```
int main() {
   BoardPosition a {5, 4};
   if (true) {
      BoardPosition b {2, 1};
      BoardPosition *c = new BoardPosition{4, 5};
      unique_ptr<BoardPosition> up = std::make_unique<BoardPosition>(9, 9);
   }
}
```

   (a) How many `BoardPosition` objects are constructed in this example?

   (b) How many `BoardPosition` objects are destructed by the time `main` ends, but before the operating system closes the program?

   (c) Is `c` a `BoardPosition` object? Why or why not?

2. Answer True or False to the following questions:

   (a) You can declare a pointer to point to a value on the heap.

   (b) You can declare a pointer to point to a value on the stack.

   (c) It is safe to delete a pointer when it points to a value on the heap.

   (d) It is safe to delete a pointer when it points to a value on the stack.

   (e) A function can determine at run-time whether a pointer points to a stack or heap value.

   (f) It is safe to blindly delete any pointer.

3. In your own words, describe the differences between the three places "const" can appear in a function prototype. You can refer to the following prototype as an example:
   `const Deck& Deck::DoSomething(const Deck &parameter) const;`

4. 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.

How to Get Credit

See above.