Chapter 12 Review Questions

1. a. The default constructor never initializes the pointer.

   b. This is simply storing the address of the string in `str`, not the string itself.

   c. No memory has been allocated to `str`. The constructor should include the code

   ```
   str = new char[len+1]
   ```

   Note that 1 needs to be added to `len` to account for the trailing zero.

2. First, you need to remember to delete any memory allocated by new in the class destructor. Second, initializing one object to another copies pointer values but not the actual data. You need to create a copy constructor that allocates space using `new` and then copies the actual data (not just the pointer) to that space. Third, assigning one object to another will cause a similar problem. Similarly, the solution is to define an assignment operator that allocates space using `new` and then copies the actual data (not just the pointer) to that space.

3. The compiler will automatically generate a constructor if you don't provide one explicitly. It takes no objects and does nothing. The compiler will also automatically generate a copy and assignment constructor which use memberwise assignment. It will also generate a default destructor which takes no objects and does nothing.

4. `Personality` must be declared as a `char array`. The constructor `nifty:nifty (char *s)` does not account for the trailing zero when allocating memory using `strlen`. It also does not copy the data to `personality`, it merely copies the address `*s` points to.

5. A1. The default constructor Golfer()
   A2. The copy constructor Golfer (const & g)
   A3. Golfer(const char * name, int g = 0)
   A4. The default constructor Golfer()
   A5. The copy constructor Golfer (const & g)
   A6. Golfer(const char * name, int g = 0)
   A7. Default assignment operator (which is not defined)
   A8. Default assignment operator (which is not defined)

   b. The class needs an assignment operator that allocates space using `new` and then copies the actual data (not just the pointer) to that space.