Copy Constructor Overview Solutions

Copy Constructor Summary

- Briefly explain what a copy constructor is
  - The copy constructor is a specialized form of constructor which is used to initialize a new object from an existing object of the same class
  - It is used to pass arguments and return values into and out of a function call by value
- What is the prototype of the copy constructor?

```
T(const T& other);          // Copy constructor for type T
```

Copy Constructor and Variable Creation

- How is the copy constructor invoked?
  - The copy constructor is automatically invoked when we create an object with an initial value

```
Test test1(x, y);           // Calls Test's constructor (not the copy constructor!)
Test test2{test1};          // Create Test object test2 and initialize it from test1
Test test3 = test1;         // Create Test object test3 and initialize it from test1
```

  - The copy constructor is also invoked when passing to and returning from a function by value

When to Write a Copy Constructor

- Explain why it is not normally necessary to implement a copy constructor when writing a class
  - If we do not provide a copy constructor, the compiler will synthesize a copy constructor which
    - Copies data members which are built-in types
    - Calls the copy constructor of members which are classes
- In what circumstances is it necessary to implement a copy constructor?
  - When the default is not good enough
  - Usually this is when the class manages a resource

Example of default copy constructor

- The following class does not define a copy constructor
- Write an example of the code that the compiler could generate when synthesizing a copy constructor

```
class Test {
   int i;
   string str;
public:
   ...   // Class interface function declarations
   // Compiler-generated copy constructor
   // Initializes "this" by copying the argument's i member
   // and calling std::string's copy constructor for y
   // Test(const Test& other) : i(other.i), y(other.y) {}
};
```