1. Create a **class Counted** that contains an **int id** and a **static int count** . The default constructor should begin: **Counted ( ) : id(count++)**. **{**It should also print its **id** and that it’s being created. The destructor should print that it’s being destroyed and its **id**. Test your class.

2. Prove to yourself that **new** and **delete** always call the constructors and destructors by creating an object of **class Counted** (from Exercise 1) with **new** and destroying it with **delete**. Also create and destroy an array of these objects on the heap.

3. Create a **PStash** object and fill it with **new** objects from Exercise 1. Observe what happens when this **PStash** object goes out of scope and its destructor is called.

4. Repeat Exercise 4, but add a member function **f( )** to **Counted** that prints a message. Move through the **vector** and call **f( )** for each object.

5. Dynamically create an array of objects of **class Counted** (from Exercise 1). Call **delete** for the resulting pointer, *without the square brackets*. Explain the results. 9. Create an object of **class Counted** (from Exercise 1) using **new**, cast the resulting pointer to a **void\***, and delete that. Explain the results.

6. Create a class with an overloaded operator **new** and **delete** both the single-object versions and the array versions. Demonstrate that both versions work.

7. Create a class with a placement **new** with a second argument of type **string**. The class should contain a **static vector<string>**where the second **new** argument is stored. The placement **new** should allocate storage as normal. In **main ( )**, make calls to your placement **new** with **string** arguments that describe the calls (you may want to use the preprocessor’s **\_\_FILE\_\_** and **\_\_LINE\_\_** macros).