As minimal preparation for Quiz C, review:

- Chapter 6 in your textbook
- The Chapter 6 section of the COP 2000 Glossary (on the class web site)
- The Chapter 6 Example pages (on the class web site)

Also be prepared to answer questions about the use of the & operator (as described in chapter 6) to declare a reference variable (parameter).

You will be asked questions related to C++ syntax errors. The statements in the questions below provide examples of some typical errors. (Assume that any variable mentioned has been declared in advance.)

SOURCE: `void MyFunc (int A)   // Function prototype for MyFunc`
ERROR: Function prototype statements must be terminated with a semicolon.

SOURCE: `void MyFunc (int A, B);  // Function prototype for MyFunc`
ERROR: The second formal parameter (B) is missing its data type.

SOURCE: `MyFunc (int N);  // Call function MyFunc passing it integer N`
ERROR: Identifiers are not declared in calling statements – remove the "int".

SOURCE: `MyFunc (123);  // Call function MyFunc passing it an integer constant`
ERROR: None – as long as the function is declared with an integer formal parameter to receive the value..

Know how to write and recognize valid C++ source code to perform each of the following tasks:

- declare a function prototype (w/semicolon) for a function that was coded following the function that calls it
- define and then call a function that will not return any data (should have a void data type) and receives no data when called (should have an empty formal parameter list and an empty actual parameter list)
- define and then call a function that returns one value (through the function label) and receives no data when called (should have an empty formal parameter list and an empty actual parameter list)
- define and then call a function that does not return any data, but which expects to receive data when called (should have a non-void formal parameter list with data types and an appropriate actual parameter list)
- define and then call a function that returns one value (through the function identifier) and which expects to receive data when called (should have a non-void formal parameter list with data types and an appropriate actual parameter list)
- define and then call a function that will pass more than one value back to variables declared by the parent function. The child function should be declared as having a void data type and use reference variables as the formal parameters passed in to be used as aliases for the identifiers in the parent function.
- call a function that was declared as having a void data type (does not return any data to its parent function) and expects to receive data when called

Identifier Scope Issues:

The table on the following page contains C++ source code that passes data between a variety of functions. Some of the data is "passed by value". Other data is "passed by reference". For practice in recognizing various identifiers and their scope, try to fill-in the table on the right side of the code. Then compare your answer to the ones on the next page. If you have questions, contact your instructor. In the "Scope" column, enter either "Global" or name the function in which the identifier is local.

Source Code

```
#include <iostream>
using namespace std;

double AREA; // Position #1 in the code

void Intro ()
{ cout << ("Area Calculator\n\n"); }

void GetLW (float &L, float &W)
{
 cout << ("Length? "); cin >> L;
 // Position #2
 cout << ("Width? "); cin >> W;
}

double CalcArea (float N1, float N2)
{
 AREA = N1 * N2; // Position #3 in the code
 return AREA;
}

void DispArea (double A)
{
 cout << "The area is " << A << endl;
}

int main()
{
 float LENGTH, WIDTH;
 Intro();
 GetLW (LENGTH,WIDTH);
 AREA = CalcArea (LENGTH, WIDTH);
 DispArea (AREA); // Position #4 in the code
 return 0;
}
```

Identifier	Scope
`AREA at Position #1`	
`Intro`	
`L`	
`N1`	
`A`	
`LENGTH`	

1. Would this source code compile? ____________________
2. List any identifiers that are global in scope: ____________________
3. List any identifiers being used as function names: ____________________
4. List any identifiers being used as arguments (actual parameters): ____________________
5. List any identifiers being used as formal parameters: ____________________
6. List any identifiers being used as reference variable parameters: ____________________
7. Is the identifier AREA at position #1 the same storage location as AREA at position #3? ________
8. Would a change to L at Position #2 affect LENGTH (in `main`)? ____________________
9. Would a change to N1 at Position #2 affect LENGTH (in `main`)? ____________________
10. Could a statement added to function DispArea call function Intro? ____________________
11. Could a statement added to function CalcArea call function DispArea? ____________________
12. If the return statement was removed from CalcArea and its calling statement was changed to just `CalcArea (LENGTH, WIDTH);` (without assignment to AREA), would the program still produce the same result? ____________________
13. Would the following statement be valid in main? ____________________

```
cout << "The area is " << CalcArea (LENGTH, WIDTH) << endl;
```

Answers to the Questions on the preceding page:

Source Code

```
#include <iostream>
using namespace std;

double AREA; // Position #1 in the code

void Intro ()
{ cout << ("Area Calculator\n\n"); }

void GetLW (float &L, float &W)
{
 cout << ("Length? "); cin >> L;
 // Position #2
 cout << ("Width? "); cin >> W;
}

double CalcArea (float N1, float N2)
{
 AREA = N1 * N2; // Position #3 in the code
 return AREA;
}

void DispArea (double A)
{
 cout << "The area is " << A << endl;
}

int main()
{
 float LENGTH, WIDTH;
 Intro();
 GetLW (LENGTH,WIDTH);
 AREA = CalcArea (LENGTH, WIDTH);
 DispArea (AREA); // Position #4 in the code
 return 0;
}
```

Identifier	Scope
`AREA` (at Position #1)	Global
`Intro`	Global
`L`	GetLW
`N1`	CalcArea
`A`	DispArea
`LENGTH`	main

1. Would this source code compile? Yes
2. List any identifiers that are global in scope: AREA, Intro, GetLW, CalcArea, DispArea, main
3. List any identifiers being used as function names: Intro, GetLW, CalcArea, DispArea, main
4. List any identifiers being used as arguments (actual parameters): LENGTH, WIDTH, AREA
5. List any identifiers being used as formal parameters: L, W, N1, N2, A
6. List any identifiers being used as reference variable parameters: L, W
7. Is the identifier AREA at position #1 the same storage location as AREA at position #3? Yes
8. Would a change to L at Position #2 affect LENGTH (in `main`)? Yes, it's a reference variable
9. Would a change to N1 at Position #2 affect LENGTH (in `main`)? No, it was passed by value
10. Could a statement added to function DispArea call function Intro? Yes. DispArea follows Intro
11. Could a statement added to function CalcArea call function DispArea? Not without a prototype
12. If the return statement was removed from CalcArea and its calling statement was changed to just `CalcArea (LENGTH, WIDTH);` (without assignment to AREA), would the program still produce the same result? Yes, because AREA is global. It can be accessed in all functions.
13. Would the following statement be valid in main? Yes, CalcArea produces data.

```
    cout << "The area is " << CalcArea (LENGTH, WIDTH) << endl;
```