Instructions: This homework assignment focuses primarily on C++ scope rules and functions. The answers to the following questions can be determined from Chapters 3 through 9 of the lecture notes and Chapters 2 through 8 of the text. Assume any `#include` directives, variable declarations, etc, which are needed to make the given code syntactically correct.

1) Which of the following statements concerning the scope of C++ identifiers is correct?

   1) The scope of an identifier begins at the point of its declaration and terminates at the end file containing the code.
   2) The scope of an identifier begins at the beginning of the block in which it is declared and terminates at the end of the block that contains its declaration.
   3) The scope of an identifier begins at the point of its declaration and terminates at the end of the block, if any, which contains its declaration.
   4) None of these are correct

2) If a variable `Alpha` is accessible only within a function `F`, then `Alpha` is either

   1) a global variable or a formal parameter of `F`.
   2) a local variable declared within `F` or a formal parameter of `F`.
   3) a global variable or an actual parameter passed to `F`.
   4) a local variable declared within `F` or an actual parameter to `F`.
   5) None of these

3) If the identifier `Beta` is declared as a formal parameter of a function `F`, then the scope of `Beta`:

   1) is the body of the implementation of `F`.
   2) extends from the declaration of the function `F` to the end of the file containing the code.
   3) is empty; that is, `Beta` has no scope.
   4) None of these

4) The use of global variables in a program is:

   1) good design practice and should be encouraged.
   2) a tool of lazy and weak-witted designers.
   3) an abomination in the eyes of man and god.
   4) 2 and 3 only.
   5) None of these

5) Suppose the first few lines of a function are as follows:

    ```cpp
    void Calc( float Beta ) {
       Alpha = 3.8 * Beta;
       ...
    }
    ```

    If the code compiles, then the variable `Alpha` must be:

   1) a local variable declared later in the body of `Calc()`
   2) a global variable
   3) a parameter passed to `Calc()`
   4) 1 or 2 only
   5) 1 or 3 only
   6) 2 or 3 only
   7) None of these
For questions 6 through 10, consider the following program:

```c
const int LIMIT = 50; // Line 1
int AddEm(int x, int y); // Line 2
int main() { // Line 3
    int x = 42, // Line 4
        y = 35; // Line 5
    int Sum; // Line 6
    Sum = AddEm(x, y); // Line 7
    return 0; // Line 8
} // Line 9
int AddEm(int x, int y) { // Line 10
    int Total; // Line 11
    Total = x + y; // Line 12
    if (Total > LIMIT) // Line 13
        Total = 0; // Line 14
    return (Total); // Line 15
} // Line 16
```

6) What is the scope of the identifier Sum which is declared in Line 6?
   1) Line 1 to Line 16
   2) Line 6 to Line 16
   3) Line 6
   4) Line 6 to Line 7
   5) Line 6 to Line 9
   6) None of these

7) What is the scope of the identifier x which is declared in Line 4?
   1) Line 1 to Line 16
   2) Line 4 to Line 16
   3) Line 4
   4) Line 4 to Line 7
   5) Line 4 to Line 9
   6) None of these

8) What is the scope of the identifier x which is declared in Line 10?
   1) Line 1 to Line 16
   2) Line 4 to Line 16
   3) Line 10
   4) Line 10 to Line 12
   5) Line 10 to Line 16
   6) None of these

9) What is the scope of the identifier LIMIT which is declared in Line 1?
   1) Line 1 to Line 16
   2) Line 1 to Line 3
   3) Line 1
   4) Line 10 to Line 13
   5) Line 10 to Line 16
   6) None of these

10) Which of the following are true?
    1) LIMIT is local to main() 6) All of them are true
    2) Total is local to AddEm() 7) All but 1 are true
    3) Sum is local to main() 8) 2 and 3 only
    4) LIMIT is global 9) 2, 3 and 4 only
    5) x is global 10) None of these
11) **Formal** parameters are listed in the function _________ and **actual** parameters are listed in the function _________.

1) call, implementation  
2) implementation, call  
3) header, body  
4) body, header  
5) None of these

12) When parameters are passed between the calling code and the called function, formal and actual parameters are matched according to:

1) their data types  
2) their names  
3) their relative positions in the formal and actual parameter lists  
4) whether they are inputs to or outputs from the function  
5) None of these

13) A parameter of a simple type, such as `int` or `double`, should be passed by value if that parameter's data flow is:

1) one-way, into the function.  
2) one-way, out of the function.  
3) two-way, into and out of the function.  
4) None of these

14) Which of the following statements are true when a parameter is passed by value?

1) The **actual** parameter is never modified by execution of the called function.  
2) The **formal** parameter is never modified by execution of the called function.  
3) The **actual** parameter must be a variable.  
4) All of these are false.  
5) 2 and 3 only  
6) None of these

15) Which of the following statements are true when a parameter is passed by reference?

1) The **actual** parameter can be modified by execution of the called function.  
2) The **formal** parameter can be modified by execution of the called function.  
3) The **actual** parameter cannot be a variable.  
4) All of these are false.  
5) 1 and 2 only  
6) None of these

16) Which of the following statements are true when a parameter is passed by constant reference?

1) The **actual** parameter can be modified by execution of the called function.  
2) The **formal** parameter can be modified by execution of the called function.  
3) The **actual** parameter cannot be a variable.  
4) All of these are false.  
5) 1 and 2 only  
6) None of these
17) If an ampersand ('&') is not attached to the data type of a formal parameter, then the corresponding actual parameter can be:

1) a constant  
2) a variable name  
3) an arbitrary expression  
4) All of these  
5) 1 and 2 only  
6) 2 and 3 only  
7) None of these

18) Given the function prototype and declarations:

```c
float Fix(int& N, float X);
int someInt = 10;
float someFloat = 4.3;
```

which of the following function calls would be syntactically correct?

1) `Fix(someInt, 6.85);`
2) `someFloat = Fix(24, 6.85);`
3) `someFloat = 0.3 * Fix(someInt, 6.85);`
4) `Fix(someInt + 5, someFloat);`
5) all of the above  
6) 1 and 3 only  
7) 2 and 4 only  
8) None of these

19) A function `SomeFunc` has two formal parameters, `alpha` and `beta`, of type `int`. The data flow for `alpha` is one-way, into the function. The data flow for `beta` is two-way, into and out of the function. What is the most appropriate function prototype for `SomeFunc`?

1) `void SomeFunc( int alpha, int beta );`
2) `void SomeFunc( int& alpha, int beta );`
3) `void SomeFunc( int alpha, int& beta );`
4) `void SomeFunc( int& alpha, int& beta );`
5) 1 and 2 only  
6) 3 and 4 only  
7) None of these

20) For the function definition

```c
void Func( int& Gamma ) {
    Gamma = 245;
}
```

which of the following comments best describes the direction of data flow for `Gamma`?

1) one-way, into the function  
2) one-way, out of the function  
3) two-way, into and out of the function  
4) None of these

21) For the function definition

```c
void Func( int Gamma ) {
    cout << 3 * Gamma;
}
```

which of the following comments best describes the direction of data flow for `Gamma`?

1) one-way, into the function  
2) one-way, out of the function  
3) two-way, into and out of the function  
4) None of these
22) For the function definition

```cpp
void Func( int& Gamma ) {
    Gamma = 3 * Gamma;
}
```

which of the following comments describes the direction of data flow for Gamma?

1) one-way, into the function  
2) one-way, out of the function  
3) two-way, into and out of the function  
4) None of these

23) Consider the function definition

```cpp
void Demo( int intVal, double& doubleVal ) {
    intVal = intVal * 2;
    doubleVal = double(intVal) + 3.5;
}
```

What values does the following code fragment print?

```cpp
int myInt = 20;
double myDble = 4.8;
Demo(myInt, myDble);
cout << "myInt = " << myInt << " and myDble = " << myDble << endl;
```

1) myInt = 20 and myDble = 43.5  
2) myInt = 40 and myDble = 4.8  
3) myInt = 20 and myDble = 4.8  
4) myInt = 40 and myDble = 43.5  
5) None of these

24) Consider the function definition

```cpp
void Demo( int& intVal, double doubleVal ) {
    intVal = intVal * 2;
    doubleVal = double(intVal) + 3.5;
}
```

What values does the following code fragment print?

```cpp
int myInt = 20;
float myDble = 4.8;
Demo(myInt, myDble);
cout << "myInt = " << myInt << " and myDble = " << myDble << endl;
```

1) myInt = 20 and myDble = 43.5  
2) myInt = 40 and myDble = 4.8  
3) myInt = 20 and myDble = 4.8  
4) myInt = 40 and myDble = 43.5  
5) None of these
25) In the following function, the declaration of Beta includes an initialization.

```c
void SomeFunc( int Alpha )
{
    int Beta = 25;
    ...
}
```

Which of the following statements about the variable Beta declared above is false?

1) It is initialized once only, the first time the function is called. 4) 1 and 3 only
2) It is initialized each time the function is called. 5) 2 and 3 only
3) It cannot be reassigned a different value within the function. 6) None of these are false

26) Given the function definition

```c
void SomeFunc( ... )
{
    float Alpha;
    ...
}
```

Which of the following statements about the variable Alpha declared above is false?

1) The memory allocated to Alpha is deallocated when the function returns. 4) Alpha can be accessed directly from code outside the function.
2) No parameter in the function heading can also be named Alpha. 5) None of these are false.
3) The value of Alpha is unknown at the moment control enters the function.

For questions 27 and 28, consider the short program:

```c
#include <iostream>  // Line 1
using namespace std; // Line 2
int main() { // Line 3
    int alpha = 3; // Line 4
    int beta = 20; // Line 5
    if (beta > 10) // Line 6
    { // Line 7
        int alpha = 5; // Line 8
        beta = beta + alpha; // Line 9
        cout << alpha << ' ' << beta << '; // Line 10
    } // Line 11
    cout << alpha << ' ' << beta; // Line 12
    return 0; // Line 13
} // Line 14
```

27) What is the scope of the identifier alpha declared in Line 4?

1) Line 4 through Line 14 3) Lines 4 and 5 only
2) Lines 4, 5, 6, 12, 13 and 14 only 4) None of these
28) What is the output of the given program?

1) 3 20  
2) 3 25 3 25
3) 5 25 5 25  
4) 5 25 3 25  
5) 5 25 3 20  
6) None of these

29) This question demonstrates the hazard of choosing inappropriate parameter-passing mechanisms. Given the function definition

```c
int Power(int& Base, int& Exponent) {
    int Product = 1;
    while (Exponent >= 1) {
        Product = Product * Base;
        Exponent--;
    }
    return product;
}
```

what is the output of the following code?

```
int N = 2;
int Pow = 3;
int Result = Power(N, Pow);
cout << N << " to the power " << Pow << " is " << Result;
```

1) 2 to the power 3 is 8  
2) 2 to the power 0 is 8  
3) 0 to the power 0 is 0  
4) 2 to the power 3 is 1  
5) None of these

30) This program illustrates some of the hazards of using global variables. What is the output of the following program?

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

void Try( int& a, int b );
int x, y, z;

void main( ) {
    x = 1;
    y = 2;
    z = 3;
    Try(y, x);
    cout << x << ' ' << y << ' ' << z << endl;
}

void Try( int& a, int b ) {
    int x;
    x = a + 2;
    a = a + 3;
    b = x + a;
}
```

1) 10 6 3  
2) 10 2 3  
3) 1 2 3  
4) 1 6 3  
5) None of these