

Instructions: This homework assignment focuses primarily on C++ 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) When parameters are passed between the calling code and the called function, formal and actual parameters are matched by:

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

2) A parameter as a simple type for example `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) 1 and 2 above
- 5) 2 and 3 above
- 6) none of these

3) Given the function prototype and declarations:

```
float Fix(int& , float );  
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 above
- 7) 2 and 4 above
- 8) none of the above

4) 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 above
- 6) 3 and 4 above
- 7) none of these

9) Consider the function definition

```
void Demo( int intVal, float& floatVal ) {
    intVal    = intVal * 2;
    floatVal  = float(intVal) + 3.5;
}
```

What values are printed by the following code fragment?

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

- 1) myInt = 20 and myFloat = 43.5
- 2) myInt = 40 and myFloat = 4.8
- 3) myInt = 20 and myFloat = 4.8
- 4) myInt = 40 and myFloat = 43.5
- 5) none of the above

10) Consider the function definition

```
void Demo( int& intVal, float floatVal ) {
    intVal    = intVal * 2;
    floatVal  = float(intVal) + 3.5;
}
```

What values are printed by the following code fragment?

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

- 1) myInt = 20 and myFloat = 43.5
- 2) myInt = 40 and myFloat = 4.8
- 3) myInt = 20 and myFloat = 4.8
- 4) myInt = 40 and myFloat = 43.5
- 5) none of the above

11) 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) 1 and 2 above
- 5) 1, 2, and 3 above
- 6) none of the above

12) For the function definition

```
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 the above

13) For the function definition

```
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 the above

14) For the function definition

```
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 the above

15) For the function definition

```
void Func( int& alpha, int beta ) {  
    int delta;  
  
    delta = beta;  
    alpha = beta;  
    beta = 23;  
    cout << delta * beta;  
}
```

what is the function precondition?

- 1) // Pre: alpha is assigned a value before the function is called
- 2) // Pre: beta is assigned a value before the function is called
- 3) // Pre: delta is assigned a value before the function is called
- 4) // Pre: alpha and beta are assigned values before the function is called
- 5) // Pre: alpha, beta, and delta are assigned values before the function is called
- 6) none of the above

16) If a variable alpha is accessible only within function F, then alpha is either

- 1) a global variable or a formal parameter of F.
- 2) a local variable within F or a formal parameter of F.
- 3) a global variable or an actual parameter to F.
- 4) a local variable within F or an actual parameter to F.
- 5) none of the above

17) Choose the correct statement:

- 1) A function prototype is global if it is placed outside function definitions in the source file.
- 2) A function prototype is local if it is placed in a function definition.
- 3) The scope of a global prototype begins at the point it is placed and extends until the end of the source file.
- 4) The scope of a local prototype begins at the point it is placed and extends until the end of the function in which it appears.
- 5) All of the above are correct
- 6) None of the above are correct

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

```
void Calc( float beta )  
{  
    alpha = 3.8 * beta;
```

Assuming the code compiles, then the variable alpha is

- 1) a local variable
- 2) a global variable
- 3) a parameter
- 4) none of the above

19) In the following function, the declaration of beta includes an initialization.

```
void SomeFunc( int alpha )  
{  
    int beta = 25;  
    ...  
}
```

Which of the following statements about beta is *false*?

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

20) Which of the following rules concerning scope of variables is correct?

- 1) The scope of a global variable begins at the point of its declaration and terminates at the end of the first program block that follows it.
- 2) The scope of a local variable begins at the beginning of the block in which it is declared and terminates at the end of the block that contains it.
- 3) The scope of a formal parameters is the body of their associated function definition.
- 4) All of the above are correct
- 5) None of the above are correct

21) Given the function definition

```
void SomeFunc( ... )
{
    float alpha;
    ...
}
```

which of the following statements about alpha is *false*?

- 1) The memory allocated to alpha is deallocated when the function returns.
- 2) A parameter in the function heading can also be named alpha.
- 3) The value of alpha is undefined at the moment control enters the function.
- 4) alpha cannot be accessed directly from code outside the function.
- 5) None of them is false.

22) What is the output of the following code fragment?

```
int alpha = 3;
int beta = 20;
if (beta > 10) {
    int alpha = 5;

    beta = beta + alpha;
    cout << alpha << ' ' << beta << ' ';
}
cout << alpha << ' ' << beta;
```

- | | |
|--------------|----------------------|
| 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 the above |

23) What is the appropriate function prototype for a function that receives a character letter grade and returns its integer equivalent on a four-point grading scale?

- | | |
|---------------------------|---------------------------|
| 1) void IntEquiv(char); | 2) void IntEquiv(int); |
| 3) int IntEquiv(char); | 4) int IntEquiv(char&); |
| 5) char IntEquiv(int); | 6) none of the above |

