# Section 15.2 C++

15.2 Q1: Today, virtually all new major operating systems are written in:

1. B or BCPL.
2. C or C++.
3. UNIX.
4. Smalltalk.

ANS: b. C or C++.

15.2 Q2: C++ is a:

1. Typeless language.
2. Hybrid object-oriented language.
3. Subset of the C Language.
4. Pure object-oriented language.

ANS: b. Hybrid object-oriented language.

# Section 15.3 A Simple Program: Adding Two Integers

15.3 Q1: End-of-line comments that should be ignored by the compiler are denoted using:

1. Two forward slashes ( **//** ).
2. Three forward slashes ( /// ).
3. A slash and a star ( **/\*** ).
4. A slash and two stars ( **/\*\*** ).

ANS: a. Two forward slashes ( **//** ).

15.3 Q2: A(n) \_\_\_\_\_\_\_\_ enables a program to read data from the user.

1. std::cout.
2. std::cin.
3. return statement.
4. main declaration.

ANS:b. std::cin.

15.3 Q3: Which of the following is a variable declaration statement?

1. int total;
2. #include <iostream>
3. int main()
4. // first string entered by user

ANS: a. int total;

# Section 15.4 C++ Standard Library

15.4 Q1: Which of the following statements about the C++ Standard Library is false:

1. The C++ Standard Library consists of classes and functions that perform tasks.
2. The C++ Standard Library is an important part of the C++ “world.”
3. An advantage of using classes and functions from the C++ Standard Library is saving the effort of designing, developing and testing new classes.
4. The C++ Standard Library functions and classes are not included in every C++ implementation.

ANS: d. The C++ Standard Library functions and classes are not included in every C++ implementation.

# Section 15.5 Header Files

15.5 Q1: Each standard library has a corresponding:

1. Function.
2. Variable type.
3. Header file.
4. Cd-rom.

ANS: c. Header file.

15.5 Q2: Which of the following C++ Standard Library header files does not contain a C++ Standard Library container class?

1. <vector>.
2. <list>.
3. <stack>.
4. <string>.

ANS: d. <string>.

# Section 15.6 Inline Functions

15.6 Q1: The inline keyword:

1. Increases function-call overhead.
2. Can reduce a function’s execution time but increase program size.
3. Can decrease program size but increase the function’s execution time.
4. Should be used with all frequently used functions.

ANS: b. Can reduce a function’s execution time but increase program size.

# Section 15.8: Reference and Reference Parameters

15.8 Q1: When an argument is passed-by-value, changes in the calling function \_\_\_\_\_\_\_\_\_\_ affect the original variable’s value; when an argument is passed call-by-reference, changes \_\_\_\_\_\_\_\_\_\_ affect the original variable’s value.

1. Do not, do.
2. Do not, do not.
3. Do, do.
4. Do, do not.

ANS: a. Do not, do.

15.8 Q2: A reference parameter:

1. Is an alias for its corresponding argument.
2. Is declared by following the parameter’s type in the function prototype by an ampersand (&).
3. Cannot be modified.
4. Both (a) and (b).

ANS: d. Both (a) and (b).

15.8 Q3: Call-by-reference can achieve the security of call-by-value when:

1. The value being passed is small.
2. A large argument is passed in order to improve performance.
3. A pointer to the argument is used.
4. The const qualifier is used.

ANS: d. The const qualifier is used.

# Section 15.8: Empty Parameter Lists

15.9 Q1: Which of the following is correct keyword to explicitly indicate that a function does not receive any parameters?

1. empty
2. void
3. epl
4. none

ANS: b. void

15.9 Q2: Which of the following is false about the following function prototype?

void functionA( void );

1. It does not receive any arguments.
2. It could have been written void functionA( );.
3. It does not return a value.
4. It could have been written functionA( void );.

ANS: d. It could have been written functionA( void );.

# Section 15.10 Default Arguments

15.10 Q1: In regards to default arguments, which of the following is false?

1. When an argument is omitted in a function call, the default value of that argument is automatically inserted by the compiler and passed in the function call.
2. They must be the rightmost (trailing) arguments in a function’s parameter list.
3. Default values can be constants.
4. Default values cannot be global variables or function calls.

ANS: d. Default values cannot be global variables or function calls.

15.10 Q2: If the function int volume( int x = 1, int y = 1, int z = 1 ); is called by the expression volume( 3 ), how many default arguments are used?

1. None.
2. One.
3. Two.
4. Three.

ANS: c. Two.

# Section 15.11 Unary Scope Resolution Operator

15.11 Q1: The unary scope resolution operator is used:

1. To access a global variable when a local variable of the same name is in scope.
2. To access any variable in an outer block when a local variable of the same name is in scope.
3. To access a global variable when it is out of scope.
4. To access a local variable with the same name as a global variable.

ANS: a. To access a global variable when a local variable of the same name is in scope.

# Section 15.12 Function Overloading

15.12 Q1: Which of the following does the C++ compiler not examine in order to select the proper overloaded function to call?

1. Types and order of the arguments in the function call.
2. The number of arguments in the function call.
3. The return type of the function.
4. It examines all of the above.

ANS: c. The return type of the function.

15.12 Q2: Overloaded functions must have:

1. Different parameter lists.
2. Different return types.
3. The same number of parameters.
4. The same number of default arguments.

ANS: a. Different parameter lists.

15.12 Q3: Type-safe linkage is ensured by:

1. Name mangling.
2. Calling the correct function.
3. The agreement of the arguments and parameters.
4. Specifying return types.

ANS: a. Name mangling.

# Section 15.13 Function Templates

# 

15.13 Q1: If a function’s program logic and operations are identical for each data type it could receive as argument(s) then a \_\_\_\_\_\_\_\_\_\_ should be used.

1. Overloaded function.
2. Recursive function.
3. Macro.
4. Function template.

ANS: d. Function template.

15.13 Q2: Which of the following is true of function templates?

1. All function templates begin with the keyword class.
2. Every formal type parameter is preceded by either keyword typename or template.
3. Formal type parameters act as placeholders for built-in types or user-defined types and are used to specify the types of arguments to the function, to specify the return type of the function, and to declare variables within the body of the function definition.
4. A programmer must define a separate function template for each template function specialization to be used in the program.

ANS: c. Formal type parameters act as placeholders for built-in types or user-defined types and are used to specify the types of arguments to the function, to specify the return type of the function, and to declare variables within the body of the function definition.

15.13 Q3: Given the following function template

template < class T >

T maximum( T value1, T value2 )

{

if ( value1 > value2 )

return value1;

else

return value2;

}

what would be returned by the following two function calls?

maximum( 2, 5 );

maximum( 2.3, 5.2 );

1. 5 and a type-mismatch error.
2. 5 and 5.2.
3. 2 and 2.3.
4. Two error messages.

ANS: b. 5 and 5.2.

# Section 15.14 Introduction to Object Technology and the UML

15.14 Q1: The other classes or functions that use a certain class are referred to as its:

1. Clients.
2. Data members.
3. Member functions.
4. Methods.

ANS: a. Clients.

15.14 Q2: Which of the following is not true of object-oriented design?

1. OOD takes advantage of inheritance relationships.
2. OOD encapsulates attributes and operations into objects.
3. OOD focuses on actions (verbs).
4. Each class can be used to create multiple objects.

ANS: c. OOD focuses on actions (verbs).