Capitulo 14

This activity contains 13 questions.

1. *Section 14.1 Introduction*

14.1 Q1: The relationship between function templates and function-template specializations is most similar to the relationship between:

- ◯ *Functions and return types.*
- ◯ *Classes and objects.*
- ◯ *Headers and source files.*
- ◯ *Classes and functions.*

2. *Section 14.2 Function Templates*

14.2 Q1: A difference between function-template specializations and overloaded functions is that:

- ◯ *Function-template specializations cannot accept user-defined types.*
- ◯ *Overloaded functions usually do not perform similar operations on each data type.*
- ◯ *Function-template specializations are generated by the compiler, not the programmer.*
- ◯ *Function-template specializations do not perform identical operations on each data type.*

3. *14.2 Q2: Function-template specializations:*

- ◯ *Are generated at compile time.*
- ◯ *Are not more concise than the equivalent set of overloaded functions.*
- ◯ *Are identical to macros.*
- ◯ *Have a maximum allowed number of type parameters.*

4. *Section 14.3 Overloading Function Templates*

14.3 Q1: A function template can be overloaded by:

- ◯ *Using non-template functions with a different name but the same parameters.*
- ◯ *Using other function templates with the same function name and parameters.*
- ◯ *Using other function templates with a different name but the same parameters.*

- *Using non-template functions with the same name and different parameters.*

5. *14.3 Q2: Assuming that all four of the following functions are defined, which one will be called by the function call square(23.4)?*

- *template< typename T1, typename T2 > T1 square(T1 num1, T2 num2).*
- *double square(double num).*
- *int square(int num).*
- *template< typename T > T square(T num).*

6. *Section 14.4 Class Templates*

14.4 Q1: Class templates:

- *Must include template< typename Type > inside the class definition.*
- *May include the statement template< typename Type > anywhere.*
- *Have the option of including the optional statement template< typename Type >.*
- *Must put template< typename Type > before the class definition.*

7. *14.4 Q2: For a class template, the binary scope resolution operator (::) is needed:*

- *Only if multiple class-template specializations will be created from this class template.*
- *Both in the prototype and definition of a member function.*
- *Only in the definitions of the member functions defined outside the class.*
- *In neither the definition nor prototype of member functions.*

8. *14.4 Q3: Function templates:*

- *Do not need a separate template< typename type > statement if they take objects from a template class as a parameter.*
- *Can include objects of template classes as parameters.*
- *Do not need a separate template< typename type > statement.*
- *Must have return type T.*

9. *Section 14.5 Nontype Parameters and Default Types for Class Templates*

14.5 Q1: Nontype parameters are:

- *Specified before the angle-bracket-enclosed type-parameter list.*
- *Required for class templates.*
- *Unable to have default arguments.*
- *const.*

10. *14.5 Q2: Default type parameters are allowed only:*

- *If the class is used as a container class.*
- *If the class template also has nontype parameters.*
- *If the class template does not have any nontype parameters.*
- *As the rightmost (trailing) parameters in a template's type-parameter list.*

11. *Section 14.6 Notes on Templates and Inheritance*

14.6 Q1: Select the incorrect statement.

- *A non-template class can be derived from a class template-specialization.*
- *A class-template specialization can be used to derive a class template.*
- *A class template can be derived from a nontemplate class.*
- *A non-template class can be used to derive a class-template specialization.*

12. *Section 14.7 Notes on Templates and Friends*

14.7 Q1: Friendship cannot be declared between a class template and:

- *A member function of another class.*
- *Another class template.*
- *A global function.*
- *A class-template specialization.*

13. *Section 14.8 Notes on Templates and static Members*

14.8 Q1: Which of the following is false?

- *With a non-template class, one copy of a static data member is shared among all*

objects created from that class.

- ◯ *One copy of each static member function is shared between all class-template specializations in the class template.*
- ◯ *Each class-template specialization created from a class template has its own copy of each static data member.*
- ◯ *static data members of both template and non-template classes are initialized at file scope.*

Clear Answers / Start Over Submit Answers for Grading

Answer choices in this exercise appear in a different order each time the page is loaded.