Quiz 5 Type Traits, Template Template Parameters and Policies

Summary and Goals

The quizzes in this (advanced) section test your knowledge of C++11 type traits and they are meant to complement the corresponding library `<type_traits>`. We also introduce some other related design techniques such as *policies*, *policy-based design* and *template template parameter* that will be elaborated in Modules 5 and 6.

The goal at the moment is to get a high level overview of important metaprogramming and design concepts for C++. See “Modern C++ Design” by Andrei Alexandrescu Addison-Wesley 2001 for a good background reader.

1. Give the top two advantages and applications of type traits:
   a) Creating type-independent code.
   b) “Compile-time” *reflection*.
   c) It is a replacement for subtype polymorphism.
   d) It is used to add properties to C++ types.

2. Consider the code:

```
// Testing arithmetic types
std::cout << "int*: " << std::boolalpha
             << std::is_arithmetic<int*>::value;

std::cout << "std::complex<double>: " << std::boolalpha
             << std::is_arithmetic<std::complex<double>>::value;

std::cout << "char: " << std::boolalpha
             << std::is_arithmetic<char>::value;

std::cout << "std::bitset<8>: " << std::boolalpha
             << std::is_arithmetic<std::bitset<8>>::value;
```

   What is the output?
   a) `false,false,false,false.`
   b) `false,false,true,false.`
   c) `false,true,false,true.`
   d) `false,true,true,true.`

3. What is a *template template parameter*?
   a) It is a template parameter that is itself a template class.
   b) It is a default template parameter in a template class specification.
   c) It is the inner template parameter in the declaration of a nested template class.
   d) It is the same functionality as a *variadic* template.

4. What are the uses/advantages of *template template parameter*?
   a) Their use reduces the amount of compiler-generated code.
   b) Their use reduces the amount of user-generated code.
   c) It can be used to specify *policies*.
   d) It is useful when modelling template classes with an *extra level of indirection*.

5. What is a *policy*?
   a) It is similar to a *Strategy* pattern.
   b) It refers to pure virtual member function.
   c) It is a templated data member in a class.
   d) It defines a class interface or a class template interface.

6. What is a *policy class* and which statements are true?
   a) It is a standalone reusable class that implements a policy.
   b) It is a reusable class that implements a policy that is *embedded* in other classes.
   c) Policies and hence policy classes are *syntax oriented*.
   d) Policy classes must respect the interfaces defined by their policy.

7. How can policies be implemented in C++?
   a) Class template parameters.
   b) Template template parameters.
   c) Using inheritance and subtype polymorphism.
   d) Template member functions.