COSC1076 | Advanced Programming Techniques

Revision Questions | Week 10

These are self-revision questions, to help you track if you are understanding the weekly course content.

You should FIRST answer these questions using "pen-and-paper". Only after this should you test your answers by writing and compiling programs.

1. What is a:
   (a) Strongly typed language?
   (b) Weakly typed language?
   (c) Static typed language?
   (d) Dynamic typed language?
2. In the following program, what will the type of each variable resolve to?

```cpp
#include <string>

int main(void) {
    auto a = -8;
    auto b = 6.8;
    auto c = 10.0;
    auto d = &a;
    auto e = &c;
    auto f = "hello world";
    auto g = std::string("hello world");
}
```

3. What are the requirements when overloading a function or method?
4. What is the difference between method *overloading* and method *overriding*?
5. From how many base classes may a class inherit?
6. The following question uses the below class declarations. You may assume that the implementation of these classes is correct and follows standard C++ practices.

```cpp
class A {
public:
    A();
    virtual ~A();

    virtual foo();
};

class B : public A {
public:
    B();
    virtual ~B();

    virtual foo();
};

class C : public C {
public:
    C();
    virtual ~C();

    virtual foo();
};
```

(a) For each variable in the following code-snippet, in what order will the constructors be called?

```
A* a1 = new A();
B* b1 = new B();
C* c1 = new C();
```

(b) Using the variables in the above code-snippet, which of the assignments in the below code-snippet are permitted (that is, will compile and run)?

```
A* a2;
B* b2;
C* c2;

a2 = a1;
a2 = c1;

b2 = a1;
b2 = c1;

c2 = a1;
c2 = b2;
```

(c) In the code-snippet below, which version of the `foo` method will actually be called?

```
c1->foo();
a1->foo();

A* a3 = b1;
a3->foo();
```

(d) In what order will the deconstructors of the objects be called in the below code snippet?

```
delete c1;
```

7. The following function is generic. What properties must a type have for it to be instantiated to the generic type `T`?

```
template <typename T>
T bar(T value) {
    T retVal = value;
    retVal = retVal + 1;
    return retVal;
}
```

8. What is the different between *polymorphism* and *typecasting*?
9. What is the difference between a C++ `static_cast` and `dynamic_cast`?
10. In the code-snippet below, which version of the `foo` method will be called? (This use the classes, `A`, `B` and `C` in question 6 above.)

```
A* a4 = dynamic_cast<A*>(c1);
a4->foo();
A a5 = dynamic_cast<A>(*c1);
a4->foo();
C* c6 = dynamic_cast<C*>(a1);
c6->foo();
```