Software Development with C++ – Exam Revision

Multiple Choice – 5 Questions

- Circle the correct answer, there is one answer per question. Each question is worth 1 mark.

1. Which of the following would be a valid functor declaration that takes two floats and returns an integer?
   a) int functor() (float a, float b);
   b) int operator() (float a, float b);
   c) int operator*() (fl
   d) oat a, float b);
   e) int object() (float a, float b);

2. How do you cast complex objects to their base class in C++?
   a) static_cast<ChildClass>(new ChildClass{})
   b) dynamic_cast<ChildClass *>(new BaseClass{})
   c) dynamic_cast<BaseClass *>(new ChildClass{})
   d) dynamic_cast<BaseClass>(ChildClass{})

3. How do you inherit public members from a base class in C++?
   a) public ChildClass implements BaseClass
   b) public ChildClass : BaseClass
   c) public class ChildClass : public BaseClass
   d) public class ChildClass : BaseClass

4. What of the following are correct iterator types?
   a) Input, Forward, Bidirectional, Random access iterators
   b) Backward and forward iterators
   c) Output, Input, Forward, Backward, Bidirectional, Random access iterators
   d) Output, Input, Forward, Backward, Bidirectional, Random access iterators

5. What of the following are creational design patterns?
   a) Creation, Adapter, Factory
   b) Adapter, Proxy, Type-Object, Singleton
   c) Prototype, Builder, Factory, Decorator
   d) Builder, Factory, Singleton, Prototype

Short Answer – 5 Questions

1. What are the components of the builder design pattern?

2. When and why would you use a smart pointer over a bare pointer? Provide an example.

3. What is the difference between a reference and a pointer?

4. What problem does the Type-Object design pattern help to solve?

5. List and briefly describe 2 variable storage methods available in C++17.

Output – 3 Questions

1. What is the output of the following code if a = 1 and b = 2?

```
auto lambda = [&]() {
  cout << "Hi" << endl;
  cout << a + b << endl;
  a = a + b;
};

lambda();
lambda();

cout << a << endl;
cout << b << endl;
```

2. What is the output of the following code?

```
 int x[5] = {50, 100, 150, 200, 250};
cout << *x << endl;
++(*x);
*(x + 2) = 1000;
for (int i{0}; i < int(sizeof(x) / sizeof(x[0])); ++i)
{
    cout << x[i] << endl;
}
```

3. What is the value of _a, _b, _c, _d, _e of the following code given the three class definitions below?

```
class Foo {
public:
    Foo() : _a{1}, _b{1}, _c{5} { }
    int Bar() {
        return _a + _b;
    }
protected:
    int _a;
    int _b;
    int _c;
};
class Bar {
public:
    Bar() : _d{3}, _e{2} { }
    int Foo() {
        return _d + _e;
    }
protected:
    int _d;
    int _e;
};

class FooBar : public Foo, public Bar {
public:
    FooBar() : Foo::Foo{}, Bar::Bar{} { }
    int Bar() {
        int b = _a;
        int a = _b;
        [=]() mutable {
          a = b;
        }();
        _a += a;
        _b += b;
        _e = Foo::Bar();
        _c = Bar::Foo();
        return 1;
    }
};
int main() {
    FooBar *barfoo = new FooBar();
    barfoo->Bar();
    delete barfoo;
}
```

Code – 2 Questions

1. Write a class structure that fits the scenario below. Methods should be declared and defined in the same class. There is no need to write a header file then a code file.

Animals all have nice collars. Collars are different depending on the animal type. Each collar has a bell of some sort that will return a string representing the way they ring. There are a few different types of animals: cats, dogs and cows. Cats have a Small Collar, dogs have a Big Collar and cows have a Giant Collar. Collars also have the name of the animal on them, animals do not have a name but do have a string description of the animal in general (you need to make this up). Animals can all be petted, if they are, they return various strings representing their reactions.

2. Using the decorator pattern for the scenario below you are to create the relevant classes and demonstrate after how you would create a Computer with 3 CPU's and 2 RAM and display the price. Once again, there is no need to create a separate file for declarations and definitions. Define in the same place as you declare.

Computers all have certain parts in them. We will focus on two of these parts: RAM and CPU. Forget how a computer is built and focus on these two parts. Computers can have various different amounts of RAM in GB and CPU core counts. We'll leave the concept of cores and GB out of this. What you need to know is that a computer can have any amount of both or none of one and some of the other. Using the decorator pattern, you need to make this possible.

All parts have a name and an attached price. Prices on all parts start from $10 for installation. RAM costs $50 each and CPU's cost $150 each. All computers have a name and a price.

Hint: draw the UML if it helps you understand the scenario. You will not be given extra marks for doing this though.