C++ test

1-Which is the output of the following programs? (justify your answer as much as possible)

PROGRAM 1:

```
class Class1
{
public:
    Class1(int n = 0) : m_n(n) { }

public:
    virtual int foo() const { return m_n; }
    virtual ~Class1() { }

protected:
    int m_n;
};

class Class2
    : public Class1
{
public:
    Class2(int n = 0) : Class1(n) { }

public:
    virtual int foo() const { return m_n + 1; }
};

int main()
{
    const Class1 obj1(1);
    const Class2 obj2(3);
    const Class1 *obj3[2] = { &obj1, &obj2 };
    typedef std::vector<Class1> vecClass;
    vecClass vec1({ obj1, obj2 });
    vecClass::const_iterator it = vec1.begin();

    std::cout << obj3[0]->foo()
              << obj3[1]->foo()
              << it->foo()
              << ( it + 1)->foo()
              << std::endl;

    return 0;
}
```

PROGRAM 2:

```
class Class1
{
public:
    Class1() : m_i(0) { }

protected:
    int m_i;
};

class  Class2
{
public:
    Class2() : m_d(0.0) { }

protected:
    double m_d;
};

class Class3
    : public Class1
    , public Class2
{
public:
    Class3() : m_c('a') { }

private:
    char m_c;
};

int main()
{
    Class3 obj1;
    Class2 *obj2 = &obj1;
    Class1 *obj3 = &obj1;

    const int n1 = (obj3 == &obj1) ? 6 : 5;
    const int n2 = (obj2 == &obj1) ? 4 : 3;
    const int n3 = (reinterpret_cast<char*>(obj3) == reinterpret_cast<char*>(obj2)) ?
2 : 1;

    std::cout << n1 << n3 << n2 << std::endl;

    return 0;
}
```

2-Implement the needed things to not use the sharedData from different threads at the same time. Please, modify the following code as you need:

```
struct CriticalData{
  SharedData shared;
};

void deadLock(CriticalData& a, CriticalData& b){
  a.shared.use();
  std::this_thread::sleep_for(std::chrono::milliseconds(1));
  b.shared.use();
}

int main(){

  CriticalData c1;
  CriticalData c2;

  std::thread t1([&]{deadLock(c1,c2);});
  std::thread t2([&]{deadLock(c2,c1);});

  t1.join();
  t2.join();
  return 0;
}
```

3-We are the middle-ware for 2 different components and we need to implement a function to make a request from one to the other, get the answer, and process it. You can define the interfaces as you want. The info type gotten does not matter. If possible, include a unit test for the class implemented (you can use gtest, trompeloeil & catch2, or any other framework). Classes to be implemented: interface with component 1, middle class, Interface with component 2, unit test. Hint: The answer from the component 2 is not immediate. Format for answers: c++ files.