Homework 04

ECE 449/590, Fall 2022

Due Date: 11/30 by the end of the day (Chicago time)

1. *(30 points)* Consider the following piece of code. For each definition of the class `A`, determine if there will be any compiling error. If so, find the first line with the error and explain why.

```
void some_function()
{
    A a;      // line 1
    A b = a;  // line 2
    b = a;    // line 3
}
```

   A.
```
class A
{
    const int member;
};
```

   B.
```
class A
{
    int member;
    ~A();
};
```

   C.
```
class B
{
    B &operator=(const B &);
};
class A
{
    B member;
};
```

2. *(25 points)* Read the following program and decide what among the three members – default constructor, copy constructor, and `operator=`, should be synthesized or implemented in type `T` for the lines 1 to 5 to be compiled correctly. For example, the line 0 requires none of the three.

```
T create(size_t n);
bool condition_one(T t);
bool condition_two(const T &t);
void modify(T &t);

T generate(size_t n) {
    T a(1);                     // line 0
    T b = create(n);            // line 1

    while (!condition_one(b)) { // line 2
        modify(b);              // line 3
        if (condition_two(b))   // line 4
           break;
    }

    return b;                   // line 5
}
```

3. *(20 points)*

   Consider the following class definitions.

```
class base {
public:
    base(bool th) {
        std::cout << "ctor of base" << std::endl;
        if (th) {
            throw std::runtime_error("throw from ctor of base");
        }
    }
    ~base() {
        std::cout << "dtor of base" << std::endl;
    }
};

class member {
public:
    member(bool th) {
        std::cout << "ctor of member" << std::endl;
        if (th) {
            throw std::runtime_error("throw from ctor of member");
        }
    }
    ~member() {
```

```
        std::cout << "dtor of member" << std::endl;
    }
};

class derived : public base {
    member m_;
public:
    derived(bool base_th, bool member_th)
        : base(base_th), m_(member_th) {
        std::cout << "ctor of derived" << std::endl;
    }
    ~derived() {
        std::cout << "dtor of derived" << std::endl;
    }
};
```

Determine the outputs of the following functions and explain why.

1)
```
void test_1() {
    try {
        derived d(false, false);
    }
    catch (std::exception &e) {
        std::cout << e.what() << std::endl;
    }
}
```

2)
```
void test_2() {
    try {
        derived d(false, true);
    }
    catch (std::exception &e) {
        std::cout << e.what() << std::endl;
    }
}
```

3)
```
void test_3() {
    try {
        derived d(true, false);
    }
    catch (std::exception &e) {
        std::cout << e.what() << std::endl;
    }
}
```

4)
```
void test_4() {
      try {
          derived d(true, true);
      }
      catch (std::exception &e) {
          std::cout << e.what() << std::endl;
      }
  }
```

4. *(25 points)* Compile and execute `smart_pointer.cpp`. Explain the output.