第一次

1. A，ls -a 可以列出所有文件和文件夹
2. AB，只有 A B 能够正确生成可执行文件
3. C，extern 后 x 为全局变量，foo 没有改变 x 的值
4. C，Max 操作带来了两次++
5. A
6. C 或 D 都算对（g++ 结果是 C，msvc 是 D）
7. BD
8. ABC，复杂的函数体不推荐使用内联函数

第二次

1. AC
2. B
3. A
4. AD
5. D
6. C
7. A
8. AD

第三次

1. ABCD

   解析：以上四个成员函数/运算符，当用户没有显式定义时，编译器都会根据自身需要自动合成。

2. AB

   解析：

   A. 保护成员允许在派生类成员函数中被访问，但不能被外部函数访问。

   B. 若想让某成员能被派生类的对象访问，可在派生类 public 部分用关键字 using 声明它的名字。

   C. 基类的私有成员对派生类私有。

   D. 赋值运算符、友元函数也不可以被派生类继承。

3. AB

   解析：

   C. 不一定，如果传入的实参为右值且该类显式定义移动构造函数，则调用移动构造函数

   D. 右值引用不能绑定左值

4. C

   解析： 子对象在进入构造函数体之前完成初始化。

   如果调用拷贝构造函数且没有给类显式定义拷贝构造函数，编译器将提供“隐式定义的拷贝构造函数”，递归调用所有子对象的拷贝构造函数。`Animal` 类自动生成的拷贝构造函数会调用 `Identity` 类定义的拷贝构造函数。

   先完成子对象构造，再完成当前对象构造，子对象构造的次序仅由在类中声明的次序所决定，析构函数的次序与构造函数相反。

5. BC

   解析：

   A. 不会发生编译错误，x 是右值引用，本身是左值

   D. (4)会发生编译错误，因为不存在 f(const int&)与其匹配

6. BC

   解析：

```
#include<iostream>
using namespace std;


class Appearance {
   int weight = 0;
public:
   Appearance(){cout << "A";}
   Appearance(int _w):weight(_w){cout << "A(" << weight << ")";}
   ~Appearance(){cout << "a";}
};
```

```
class Small{
protected:
   Appearance A;
};

class Animal {
protected:
   Appearance A;
public:
   Animal() { cout << "N"; } // (1)
   Animal(int _w){ // (2)
        cout << "N(" << _w << ")";
        A = Appearance(_w);
  }
   ~Animal() { cout << "n"; }
};

class Rabbit: public Animal { //(3)
Animal N{1};
public:
   Rabbit() { cout << "R"; }
   Appearance getA() {return A;}
   ~Rabbit() { cout << "r"; }
};

int main() {
   Rabbit R;
   return 0;
}
```

输出结果为：ANAN(1)A(1)aRrnana

A. Appearance 被实例化了 3 次，依次为，继承自 Animal 的数据成员 A，Rabbit 的数据成员 Animal 初始化数据成员 A，Rabbit 的数据成员 Animal 构造函数实例数据成员 A(1)。

B. R 不可以访问数据成员 N 的 Appearance。

C. (1)、(2)被分别执行了一次，先执行基类的构造函数来初始化继承来的数据，再初始化派生类数据，执行派生类的构造函数。

D. 由于 Appearance 产生了二义性，程序报错。

7. B

解析：B. int 和 enum 类型可以就地初始化

8. AD

解析：

B. using 关键字可以恢复被屏蔽的基类成员函数。

C. (4)是对(1)的重写覆盖，(1)无法直接被 r 调用，会报错。

D. `a.move(1)`会调用基类的 `move(int i)`函数。

第四次

1. 答案：C

解析：override 只起到语法检查的作用，A 错；(1) 是重写隐藏，B 错；(2) 处因 `Base::func1()`不是虚函数，无法实现晚绑定，C 对；Base 对象中还包含虚函数指针，因此为 12 个字节

2. 答案：A

解析：纯虚析构函数必须指定函数体，B 错；Derived2 中依旧有虚函数表指 针，C 错；析构函数中虚机制不起作用，D 错。

3. 答案：ABC
4. 答案：D

解析：纯虚函数也可以定义函数体，A 错；只有除纯虚析构函数外的其他纯虚函数被重写覆盖了，才不是抽象类，B、C 错。

5. 答案：B

解析：Base 类必须含有虚函数，才可以使用 dynamic_cast，A 错；若定义了相应的构造函数，在无继承关系的类之间也可以转换，C 错；未定义相应的构造函数，D 错

6. 答案：AB 或 ABC
解析：严格来讲本题 C 选项错误，因为编译器可能会做一些优化，把一些编译时就能确定的虚函数调用提前绑定下来。可以参考 https://www.zhihu.com/question/491602524 。但是由于我们课件中没讲，所以这次就都算对

7. 答案：BCD

解析：(1)处返回值将进行强制类型转换，A 错

8. 答案：D

解析：多重继承可以继承多个非抽象类，建议只继承一个非抽象类，A 错；构造函数不能也不必是虚函数，B 错；