2025第三次作业（选择题）

选择题

刷新

如无特殊说明，所有题目的编译选项都包含 `-std=c++11`

1. 【多选题】关于下面的程序，以下哪些选项是**正确**的？

```
#include <iostream>

class Test
{
private:
    int printed_times;
    static int class_printed_times;
public:
    const void print0();
    void print1() const;
    void print_printed_times();
    Test(/* args */);
    ~Test();
};

void Test::print_printed_times()
{
    std::cout << printed_times << std::endl;
}

const void Test::print0()
{
    std::cout << "Printing_0..." << std::endl;
    printed_times++;
}

void Test::print1() const
{
    std::cout << "Printing_1..." << std::endl;
    printed_times++;
}

Test::Test(/* args */)
{
    printed_times = 0;
}

Test::~Test() {}

int main(int argc, char const *argv[])
{
    Test test;
    test.print0();
    test.print1();
    test.print_printed_times();

    return 0;
}
```

A. `Test::print0` 是常量成员函数

B. `Test::print1` 是常量成员函数

C. 该程序不能正常编译运行，原因（之一）是没有正确地为该类的静态成员变量赋初值

D. 该程序不能正常编译运行，原因（之一）是常量成员函数不能修改非静态成员变量的值

2. 【多选题】关于以下的程序不能通过编译的原因，描述**错误**的是（）

```
#include <iostream>

class Test
{
    Test operator=(Test) = delete;
    static int count;
public:
    Test();
    ~Test();
};

int Test::count; //#2
Test::Test() { count ++; }
Test::~Test() { count --; }

int main()
{
    Test test0;
    Test test1 = test0; //#0
    std::cout << test1.count << std::endl; //#1

    return 0;
}
```

A. #0处报错，因为编译器不会再合成默认拷贝构造函数

B. #0处报错，因为默认拷贝赋值运算符 `operator=` 被禁用

C. #1处报错，因为静态成员count为类Test而非对象test1的成员

D. #1处报错，因为#2处并未给静态成员count赋初值

E. #1处报错，因为此处没有权限访问成员count

3. 【多选题】以下关于面向对象编程和类的继承的说法**正确**的是（）

A. 在private继承中，派生类的成员函数无法访问基类的public成员

B. 在public继承中，派生类的成员函数无法访问基类的private成员

C. 在private继承中，在对相应成员（于派生类的public部分）使用using关键字声明以后，外部函数可以通过派生类对象访问基类public成员

D. 在public继承中，在对相应成员（于派生类的public部分）使用using关键字声明以后，外部函数可以通过派生类对象访问基类private成员

4. 【多选题】选出**正确**的选项

```
#include <iostream>

int f(int& x)
{
    return x;
}

int f(int&& x)
{
    x++; //(0)
    int& y = x; //(1)
    const int& z = 1; //(2)
    int& w=2; //(3)
    return z;
}

int main(int argc, char const *argv[])
{
    std::cout << f(4) << std::endl;

    return 0;
}
```

A. (0)处会发生编译错误，因为右值无法编辑

B. (1)处会发生编译错误，因为左值引用无法绑定右值

C. (2)处不会发生编译错误，因为常量左值引用可以绑定右值

D. (3)处会发生编译错误，因为左值引用无法绑定右值

5. 【多选题】以下哪些描述是**正确**的?

```
class Base
{
public:
    Base(/* args */){}
    ~Base(){}
    virtual void f1(){}
    void f2(){}
    virtual void f3(){}
    virtual void f4(){}
};

class Derived: public Base
{
public:
    Derived(){}
    ~Derived(){}
    void f1(int){}
    virtual void f2(){}
    void f3(){}
};

int main(int argc, char const *argv[])
{
    Derived d;
    d.f1();
    d.f2();
    d.f3();
    d.f4();

    return 0;
}
```

A. `d.f1();` 行会报错无法编译，这是因为发生了重写隐藏，可通过在派生类中使用 `using Base::f1;` 规避

B. `d.f2();` 行发生了重写隐藏，可通过改写为 `d.Base::f2();` 规避

C. `d.f3();` 行发生了正确的重写覆盖，尽管在声明、定义派生类的f3时并未含有virtual或override关键字

D. `d.f4();` 行会报错，因为对应的虚函数在派生类中未被正确覆盖却要调用

6. 【多选题】以下哪些描述是**正确**的?

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

class Base {
public:
    virtual void print() {
        cout << "Base class" << endl;
    }
};

class Derived : public Base {
public:
    void print() override {
        cout << "Derived class" << endl;
    }
};

int main() {
    Base* base = new Base();
    Derived* derived = new Derived();

    Base* bp = dynamic_cast<Base*>(derived);
    Derived* dp = static_cast<Derived*>(base);

    bp->print();
    dp->print();

    delete base;
    delete derived;

    return 0;
}
```

A. `bp->print()` 会输出 "Base class"， `dp->print()` 会输出 "Derived class"，因为虚函数调用的识别依赖于当前指针或引用的类型。

B. `bp->print()` 会输出 "Derived class"， `dp->print()` 会输出 "Base class"，因为虚函数调用的识别依赖于当前指针或引用所指对象的类型。

C. 本段代码中若将 `static_cast` 换成 `dynamic_cast` ，程序运行中会发生段错误。

D. 本段代码中若将 `dynamic_cast` 换成 `static_cast` ，程序仍能正常运行。

7. 【多选题】以下哪些描述是**正确**的?

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

class Base {
public:
    virtual void func1() {
        cout << "Base::func1()" << endl;
    }
    void func2() {
        cout << "Base::func2()" << endl;
    }
};

class Derived : public Base {
public:
    void func1() {
        cout << "Derived::func1()" << endl;
    }
    virtual void func2() {
        cout << "Derived::func2()" << endl;
    }
};

int main() {
    Base *basePtr = new Derived();
    basePtr->func1();
    basePtr->func2();
    delete basePtr;
    return 0;
}
```

A. `func1()` 在 Derived 类中是对 Base 类的 `func1()` 的重写覆盖，尽管其在 Derived 类中的声明并未带有virtual关键字

B. `func2()` 在 Derived 类中是对 Base 类的 `func2()` 的重写覆盖，尽管其在 Derived 类中的声明并未带有virtual关键字

C. `basePtr->func1()` 调用的是 `Derived::func1()`

D. `basePtr->func2()` 调用的是 `Derived::func2()`

8. 【多选题】选出**错误**的选项:

```
#include <iostream>

class MyClass {
    int *ptr = nullptr;

public:
    void print() {
        std::cout << ptr << std::endl;
    }

    MyClass() {}

    MyClass(const MyClass &obj) {
        if (obj.ptr != nullptr) {
            ptr = new int;
            *ptr = *obj.ptr;
        }
    }

    ~MyClass() {
        if (ptr != nullptr) {
            delete ptr;
            ptr = nullptr;
        }
    }
};

int main(int argc, char const *argv[])
{
    MyClass obj;
    obj.print();

    return 0;
}
```

A. 这段代码在运行时有极大发生段错误的风险，因为并未正确为指针ptr开辟内存却要访问ptr的值。

B. 若将这段代码中，MyClass类的拷贝构造函数的定义中传入的参数类型由 `const MyClass &` 改为 `MyClass` ，尽管能通过编译，但会带来诸多风险，故并不推荐如此使用。

C. 这段代码中，MyClass类的拷贝构造函数可以防止浅拷贝的发生，避免了潜在的内存泄漏和数据访问错误。

D. 这段代码中，显式定义拷贝构造函数是不必要的，因为编译器提供的默认拷贝构造函数会正确地拷贝所有成员。

提交格式

请你提交一个文本文件，第i行是一个由A\B\C\D组成的字符串，代表第i题的答案。

若你不想提交第i题，请将第i行留空。

你提交的文本文件至少要有8行，且前8行必须由A\B\C\D组成，否则将被认为是无效提交。

你可以通过测试点的详细信息，看到评测对你提交文件的解析。若存在问题，请联系助教。

以下给出一个合法的提交答案的例子：

```
A
BC
C
A
B
C
CD
C
```

评测器对第一行的解析结果如下：

```
Valid answer: ['A']
```

注意：作业截止之前本题的评测器只检查提交格式是否正确，不检查具体答案。只要提交格式正确就显示100分。每道题的具体答案会在作业截止之后重新评测，以最后一次提交的答案为准

语言和编译选项

#	名称	编译器	额外参数	代码长度限制
0	answer	cp		1048576 B

递交历史

#	状态	时间
332244	Accepted	2025-04-18 16:10:32
328861	Accepted	2025-04-09 18:10:07

1

当前没有提交权限！