

**Instructions:** This homework assignment covers the basics of inheritance and polymorphism in C++. The answers may be determined from the CS 2704 notes and lectures and the assigned reading in Stroustrup, which are online at the course website. All inheritance relationships below are `public`.

Opscan forms will be passed out in class and collected in class on Tuesday April 16. Opscans will not be accepted at any other time. Mutilated opscans may be discarded.

---

For questions 1 through 10, suppose an inheritance hierarchy with a base class `Base` and a derived class `Derived`. Determine if each of the following statements is true or false in C++; use the answers:

- 1) true                      2) false

1. For `Derived` to override an inherited member function, `Base` must not declare the function to be virtual.
  2. If a function in `Derived` that overrides a function in `Base` is declared as virtual in `Derived`, then it is automatically virtual in `Base`.
  3. If a function is not declared at all in `Base`, then it may not be declared as virtual in `Derived`.
  4. A pure virtual function must have return type of `void`.
  5. If `Base` is an abstract class, then all function members, except constructors, of `Base` must be pure virtual functions.
  6. Virtual functions are the only C++ mechanism required to achieve runtime binding of operations to objects.
  7. In the context of a function call, “binding” refers to forming an association between the named function and a specific implementation.
  8. “Dynamic binding” refers to binding decisions that are made at compile time.
  9. If a function is declared virtual in `Base` then `Derived` must override that function.
  10. If `Base` contains a pure virtual member function, then `Derived` must provide an implementation for that function.
- 

For questions 11 and 12, suppose that a C++ class `D` is derived from a base class `B`, that class `B` has a public member function `F()`, and class `D` overrides `B::F()` with its own version. At execution time, suppose that the address of a `D` object is passed to the following function:

```
void Foo(B* x) {  
    x->F();  
}
```

11. If `F()` is declared to be virtual in class `B`, whose version of `F()` does `Foo()` call?
  - 1) `B::F()`
  - 2) `D::F()`
  - 3) Both versions are called.
  - 4) It is illegal to pass the address of a `D` here.
  - 5) None of these
12. If `F()` is not declared to be virtual in class `B`, whose version of `F()` does `Foo()` call?
  - 1) `B::F()`
  - 2) `D::F()`
  - 3) Both versions are called.
  - 4) It is illegal to pass the address of a `D` here.
  - 5) None of these

For questions 13 and 14, suppose that a C++ class D is derived from a base class B, that class B has a public member function F(), and class D overrides B::F() with its own version. At execution time, suppose that a D object is passed to the following function:

```
void Bar(B x) {
    x.F();
}
```

13. If F() is declared to be virtual in class B, whose version of F() does Bar() call?
- 1) B::F()
  - 2) D::F()
  - 3) Both versions are called.
  - 4) It is illegal to pass a D here.
  - 5) None of these
14. If F() is not declared to be virtual in class B, whose version of F() does Bar() call?
- 1) B::F()
  - 2) D::F()
  - 3) Both versions are called.
  - 4) It is illegal to pass a D here.
  - 5) None of these

For questions 15 through 30, assume the following class declarations and main(). Assume that corresponding implementations are supplied for each class.

```
class Base {
public:
    virtual void F();
    virtual void G() = 0;
    virtual void H();
    void I();
};

class D : public Base {
public:
    void F();
    virtual void G();
    void H();
    virtual void J();
};

class E : public D {
public:
    virtual void F();
    virtual void G();
    void I();
};

int main() {
    D*   pD = new D;
    Base* pB = pD;
    E*   pE = new E;

    pB->F();           // line 1
    pB->G();           // line 2
    pB->H();           // line 3
    pB->I();           // line 4

    pD->F();           // line 5
    pD->G();           // line 6
    pD->I();           // line 7
    pD->J();           // line 8

    pB = pE;
    pD = pE;

    pB->F();           // line 9
    pB->G();           // line 10
    pB->H();           // line 11
    pB->I();           // line 12

    pD->F();           // line 13
    pD->J();           // line 14

    pE->F();           // line 15
    pE->I();           // line 16
    return 0;
}
```

It is possible to answer the following questions by simply testing code. Don't do that. Try to determine the answers logically first, then check your analysis by testing if you wish. For each question, determine which function (or functions) are called when the statement in the indicated line is executed:

15. line 1

- 1) Base::F()                      2) D::F()                      3) E::F()                      4) None of these

16. line 2

- 1) Base::G()                      4) Base::H()                      7) 2 and then 4  
2) D::G()                      5) D::H()                      8) 2 and then 5  
3) E::G()                      6) 1 and then 4                      9) None of these.

17. line 3

- 1) Base::H()                      2) D::H()                      3) E::H()                      4) None of these

18. line 4

- 1) Base::I()                      2) D::I()                      3) E::I()                      4) None of these

19. line 5

- 1) Base::F()                      2) D::F()                      3) E::F()                      4) None of these

20. line 6

- 1) Base::G()                      4) Base::H()                      7) 2 and then 4  
2) D::G()                      5) D::H()                      8) 2 and then 5  
3) E::G()                      6) 1 and then 4                      9) None of these.

21. line 7

- 1) Base::I()                      2) D::I()                      3) E::I()                      4) None of these

22. line 8

- 1) Base::J()                      2) D::J()                      3) E::J()                      4) None of these

23. line 9

- 1) Base::F()                      2) D::F()                      3) E::F()                      4) None of these

24. line 10

- |              |                 |                   |
|--------------|-----------------|-------------------|
| 1) Base::G() | 4) Base::H()    | 7) 2 and then 5   |
| 2) D::G()    | 5) D::H()       | 8) 3 and then 5   |
| 3) E::G()    | 6) 2 and then 4 | 9) None of these. |

25. line 11

- |              |           |           |                  |
|--------------|-----------|-----------|------------------|
| 1) Base::H() | 2) D::H() | 3) E::H() | 4) None of these |
|--------------|-----------|-----------|------------------|

26. line 12

- |              |           |           |                  |
|--------------|-----------|-----------|------------------|
| 1) Base::I() | 2) D::I() | 3) E::I() | 4) None of these |
|--------------|-----------|-----------|------------------|

27. line 13

- |              |           |           |                  |
|--------------|-----------|-----------|------------------|
| 1) Base::F() | 2) D::F() | 3) E::F() | 4) None of these |
|--------------|-----------|-----------|------------------|

28. line 14

- |              |           |           |                  |
|--------------|-----------|-----------|------------------|
| 1) Base::J() | 2) D::J() | 3) E::J() | 4) None of these |
|--------------|-----------|-----------|------------------|

29. line 15

- |              |           |           |                  |
|--------------|-----------|-----------|------------------|
| 1) Base::F() | 2) D::F() | 3) E::F() | 4) None of these |
|--------------|-----------|-----------|------------------|

30. line 16

- |              |           |           |                  |
|--------------|-----------|-----------|------------------|
| 1) Base::I() | 2) D::I() | 3) E::I() | 4) None of these |
|--------------|-----------|-----------|------------------|