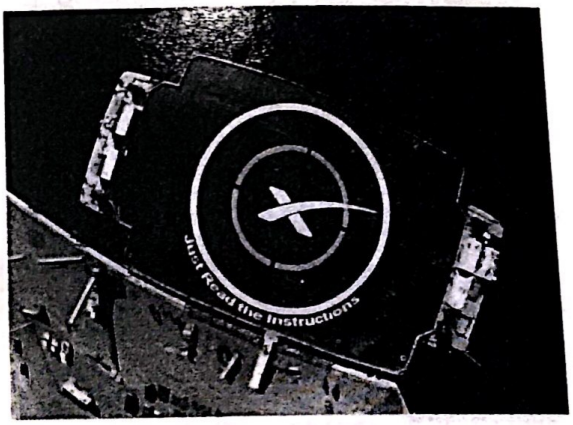
0

# CS212d Exam Two

## 2019 Spring

copy control
= copy
- assignment
- destructor

Note that I have omitted any #includes or "using namespace std;" statements in all questions, in order to save space and to save your time thinking about them. You may assume that all such statements that are needed are present. And you don't have to write them either!!!




Please, read all questions *carefully*!

Answering the short-answer questions, in particular, requires that you read and *understand* the programs shown.

If a question asks you to write a class or a function and shows you output, be sure your class / function generates that output, unless the spec states otherwise.

7/10 - 35/50

| Questions | Points |
|---|---|
| 1 | xtra |
| 2-11 | 5 |
| 12 | 50 |

50

35

Answer questions 1–11 in the exam book. For multiple choice questions, **circle** the correct answer. There should be only one correct answer / question.

Answer question 12 in your blue book.

Place your name and id on every page in this book before the end of the exam.

Name: Brodyn. NetID bb6631

3. Given:

```
class Parent {
public:
    virtual void foo() = 0;
};

class Child : public Parent {
public:
    void foo() { cout << "Child\n"; }          // Line A
};

class GrandChild : public Child {
public:
    void foo() { cout << "GrandChild\n"; }     // Line B
 };

int main() {
    GrandChild gc;                             // Line C
    Child* cp = &gc; // Since
    gc.foo(); GrandChild                       // Line D
    cp->foo(); GrandChild                      // Line E
}
```

Foo is virtual in the Base class

What will happen when we build and run the program?

a) It will fail to compile at line A, because **foo is NOT marked virtual** in Child

b) It will fail to compile at line C because GrandChild **is an abstract class.**

c) It will fail to compile at lines D and E because **foo is an abstract method.**

d) It will fail to compile for some other reason

e) It will compile, but will crash when run.

f) It will print out:
```
Child
Child
```

g) It will print out:
```
Child
GrandChild
```

h) It will print out:
```
GrandChild
Child
```

i) It will print out:
```
GrandChild
GrandChild
```

j) None of the above

Name: Burlyn NetID bxb6631

**For multiple choice questions CIRCLE the right answer**

1. **[Extra credit]** Who created C?

a) Gosling
b) Hopper
c) Ritchie C
d) Stroustrup C++
e) Thompson
f) van Rosum
g) Wall
h) None of the above

2. Given:

```
void foo(int x) {
    int* const p = &x;    // line A
    x = 17;               // line B
    cout << *p << ' ';    // line C
    *p = 28;              // line D
}

int main() {
    int y = d2;
    foo(y);
    cout << y << endl;
}
```

d=4 17 integer value can't change address p is pointing at

What is the result of compiling and running the above code? (Circle only one answer)

a) The program will have a compilation error at line A
b) The program will have a compilation error at line B
c) The program will have a compilation error at line C
d) The program will have a compilation error at line D
e) The program will have a runtme error (or undefined behavior) at line D.
f) The program will print out: 17 17
g) The program will print out: 17 28
h) The program will print out: 17 d2
i) The program will print out: d2 17
j) The program will print out: d2 28
k) The program will print out: d2 d2
l) All of the above
m) None of the above.

Name: Burlyn NetID bab631

4. Given:

```
class Foo {
public:
    Foo(string s, int n = 0) { str = s; num = n; }
    void display() { cout << str << ':' << num << endl; }
private:
    string str;
    int num;
};

int main() {
    Foo thingOne("abc", 17);
    string s = "def";
    thingOne = s;
    thingOne.display();
}
```

s is constructed as Foo("def", 0)

Implict conversion → assignment operator

What will be the result of compiling and running the program?

a. The program runs and prints:
```
abc:0
```

b. The program runs and prints:
```
def:0
```

c. The program runs and prints:
```
abc:17
```

d. The program runs and prints:
```
def:17
```

e. The program fails to compile

f. The program compiles and runs but doesn't print anything

g. The program compiles but crashes with no output

h. None of the above.

while(ifs >> x){3
ifs

- conversion operator
- copy
- assignment
- inheritance (Base = Derived) T1 a - T2 b

can make implict conversion not

explict operator bool(){3

C++ has ambiguity

if there was no default value for N) there would be a compiler error

Name: Burly2 NetID bab631

5. Given:

```
class Base {
protected:
    void protectedMethod() { }
};

class Derived : public Base { };

int main() {
    Base b;
    b.protectedMethod();  // line A
    Derived d;
    d.protectedMethod();  // line B
}
```

can't work b/c it is being called in the main()

Which of the following is true?:

a. line A will compile
   line B will compile

b. line A will compile
   line B will not compile

c. line A will not compile
   line B will compile

d. line A will not compile
   line B will not compile

protected methods can't be available outside

Name: Burlyn NetID bab631

6. Given:

```
class Pet {
public:
    virtual void eat() { cout << "Pet::eat\n"; }
};

class Cat : public Pet {
public:
    void eat() { cout << "Cat::eat\n"; }
};

int main() {
    Cat* catP = new Cat(); //new cat on heap
    Pet* petP = catP;
    petP->eat(); // polymorphism
}
```

No slicing problem b/c it is a pointer & not copy Derived instance into Base instance

What is the result of compiling and running the above program?

i. The program compiles and runs, printing "Pet::eat"

j. The program compiles and runs, printing "Cat::eat"

k. The program fails to compile because the method eat is not marked virtual in Cat.

l. The program fails to compile for some other reason.

m. The program compiles and crashes when it runs.

n. The program compiles and runs to completion without printing anything.

o. None of the above.

if eat() didn't exist in public in Pet class, then the code wouldn't work

- can't make an instance of an abstract class

- ~~can't~~ gs must provide default constructor when you have more than one constructor

Derived ptr can't be assigned to Base ptr b/c Derived is more specific than base

can't call derived behavior on a base

Name: Burkyn NetID bab631

7. What is the result of the following?

```
class Derived;  // Yes, we need this.

class Base {
public:
    virtual void method(Base& arg) {
        cout << "Base::method(Base)\n";
    }
    virtual void method(Derived& arg) {
        cout << "Base::method(Derived)\n";
    }
};

class Derived : public Base {
public:
    void method(Base& arg) {
        cout << "Derived::method(Base)\n";
    }
    void method(Derived& arg) {
        cout << "Derived::method(Derived)\n";
    }
};

void someFunc(Base& arg) {
    arg.method(arg);
}

int main() {
    Derived der;
    someFunc(der);
}
```

runtime decision b/c it is virtual

Derived Base → At compile time, it would only take in a Base

ask it if it is virtual

if not virtual, it would use Base::method(Base)

a. The program runs and prints:
`Base::method(Base)`

b. The program runs and prints:
`Base::method(Derived)`

c. The program runs and prints:
`Derived::method(Base)`

d. The program runs and prints:
`Derived::method(Derived)`

e. The program fails to compile

f. A runtime error (or undefined behavior)

g. None of the above

override keyword checks for virtual method in Base

if someFunc was passed Base by value, no polymorphism. makes a copy & arg would forever be Base

Name: Burlyn Arndall-Blake NetID bab6631

8. Given

```
class Base {
public:
    void display() { cout << "Base: " << n << endl; }
protected:
    int n = 42;
};

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

int main() {
    Base* base = new Derived();
    base->display();
}
```

What is the result of compiling and running the above code?

a) Outputs:
```
Base: 42
```

b) Outputs:
```
Derived: 42
```

c) Fails to compile because the member variable `n` is protected.

d) Fails to compile because display is marked virtual in `Derived`.

e) Runtime error (or undefined behavior)

f) None of the above.

call Base display b/c it is not virtual in Base class

1) it goes to the Derive class
2) it sees the virtual label
3) goes to where it is being over-ridden

Name: Brilyn

NetID bxb6631

10. Given

```
class FederationStarship  {
public:
    FederationStarship() {}
    void attack(string weapon) {
        cout << "FederationStarship firing " << weapon;
    }
};

class Constitution : public FederationStarship {
public:
    virtual void transport() { cout << "Beam me up!"; }
    void attack() {
        cout << "Constitution firing photon torpedos";
    }
};
```

needs to be methods as virtual to work

There is no attack method in Constitution class that can take in a string

what would be the result of:

```
int main() {
    Constitution* NCC_1701 = new Constitution();
    NCC_1701->attack("phasers");
}
```

method only public can work in your program or quality using Phasers!! attacks in Derived class

a. The program runs and prints:
   `Beam me up!`

b. The program runs and prints:
   `FederationStarship firing phasers`

c. The program runs and prints:
   `Constitution firing photon torpedos`

d. The program compiles but has a runtime error

e. Compilation error because there is no `Constitution` constructor

f. Compilation error other than (e).

g. None of the above

Name: Barlyn NetID bab631

9. Given:

```
class FlyingMachine {
public:
    FlyingMachine() {}
    void fly() { cout << "In FlyingMachine fly()"; }
};

class HangGlider : public FlyingMachine {
public:
    virtual void crash() {cout << "HangGlider crashing"; }
    void fly() { cout << "In HangGlider fly()"; }
};
```

what would be the result of:

```
int main() {
    HangGlider hanger;
    FlyingMachine flier;
    flier = hanger; //slicing problem
    flier.crash();
}
```

a. The program runs and prints:
`In HangGlider fly()`

b. The program runs and prints:
`In FlyingMachine fly()`

c. The program runs and prints:
`In HangGlider crashing()`

d. Runtime error.

e. Compilation error because hanger cannot be assigned to flier.

f. Compilation error because fly is not virtual.

g. None of the above

Flymachine has no crash method.
It's not even overriding the crash method

Name: Burlyn NetID bab631

11. Given:

```
class Member {
public:
    Member()  {cout << 'a';}
    ~Member() {cout << 'b';}
};

class Base {
    Member member;
public:
    Base()  {cout << 'c';}
    ~Base() {cout << 'd';}
};

class Derived : public Base {
public:
    Derived()  {cout << 'e';}
    ~Derived() {cout << 'f';}
};


int main() {
    Derived der;
}
```

a
c
e
f
d
b


Russian nesting doll

What is the output?

a. acebdf
b. acefdb
c. aecbfd
d. aecdfb
e. caedbf
f. caefbd
g. ceadfb
h. ceabfd
i. eacfbd
j. eacdbf
k. ecafdb
l. ecabdf
m. Fails to compile
n. Runtime error (or undefined behavior)
o. None of the above

Name: Burlyn Andall-Blake NetID bab631

**Answer question 12 in your blue book.**

12. Define a class **Skyrim**

- The class Skyrim will inherit from the class Elder.
  - Elder
    - has a constructor that takes a string representing your registration code.
    - It also has *any* necessary operators and supports copy control. You should not need to know anything more about the class.
    - NB: you are **not responsible** for defining the Elder class.
- Skyrim has two fields, the player's name and a collection of Dragon pointers. There *may be* lots of different types of Dragons, but we won't be responsible for defining those derived classes here.
- The Dragons will all be on the heap. In fact there is a Skyrim method that you are **not responsible** for writing, called add, that creates the Dragons on the heap and inserts their addresses into the collection. Note that we are not telling you anything about what parameter are provided to that method, so don't think of calling it and don't worry about providing a prototype for it.
  - NB: you are **not responsible** for defining the Dragon class.
  - Dragons support copy control, along with all necessary operators

You are responsible for defining the Skyrim class and providing the following functionality:

- A constructor taking in the player's name and registration code.
- Copy **control.** Yes, all of it. Deconstructor, assignments, copy
  - Naturally, copying should involve making a deep copy. Don't just copy pointers!
- An output operator.
  - You may choose the *format*. Obviously all of the information you have about your Skyrim instance should be included.
  - Don't worry about printing information contained in the Elder class.
- An equality operator.
  - Two Skyrim instances are considered equal if all of the *corresponding* Dragons are equal, i.e.:
    - there are the same number of Dragons
    - each Dragon in one Skyrim matches the Dragon in the same position of the collection in the other Skyrim.
  - NB, the Dragons do not have to have the same address in memory to be "equal".

Name: Burlyn Andall-Blake NetID bab631

**Answer question 12 in your blue book.**

12. Define a class **Skyrim**

- The class Skyrim will inherit from the class Elder.
  - Elder
    - has a constructor that takes a string representing your registration code.
    - It also has *any* necessary operators and supports copy control. You should not need to know anything more about the class.
    - NB: you are **not responsible** for defining the Elder class.
- Skyrim has two fields, the player's name and a collection of Dragon pointers. There *may be* lots of different types of Dragons, but we won't be responsible for defining those derived classes here.
- The Dragons will all be on the heap. In fact there is a Skyrim method that you are **not responsible** for writing, called add, that creates the Dragons on the heap and inserts their addresses into the collection. Note that we are not telling you anything about what parameter are provided to that method, so don't think of calling it and don't worry about providing a prototype for it.
  - NB: you are **not responsible** for defining the Dragon class.
  - Dragons support copy control, along with all necessary operators

You are responsible for defining the Skyrim class and providing the following functionality:

- A constructor taking in the player's name and registration code.
- Copy **control.** Yes, all of it. Deconstructor, assignments, copy
  - Naturally, copying should involve making a deep copy. Don't just copy pointers!
- An output operator.
  - You may choose the *format*. Obviously all of the information you have about your Skyrim instance should be included.
  - Don't worry about printing information contained in the Elder class.
- An equality operator.
  - Two Skyrim instances are considered equal if all of the *corresponding* Dragons are equal, i.e.:
    - there are the same number of Dragons
    - each Dragon in one Skyrim matches the Dragon in the same position of the collection in the other Skyrim.
  - NB, the Dragons do not have to have the same address in memory to be "equal".