CS2124 Exam Two
2020 Spring

Read this page **thoroughly**.

You are about to take an exam. The purpose of the exam is to fairly and accurately measure your knowledge and skill in this course's material.

Exam rules You must abide by these rules:

- The exam must represent your work alone. You may not misrepresent anyone else's work as your own. You may not submit work of which you are not the sole author.
- The exam is to be completed in isolation. During the exam, you may not communicate with any other person for any reason. This prohibition includes the use of electronic communication, such as email, texting and phone.
- The exam is to be completed without additional resources. You must formulate your answers based only on the contents of your brain. Specifically:
    - Do not use the internet.
    - Do not use other software on your computer, such as IDE, compiler, interpreter or debugger.
    - Do not consult written notes.
    - Do not consult other files on your computer, including other work you've completed for this course.
    - Do not consult books.
- Do not copy or distribute the exam document or your answers at any time. After you submit your answers delete the exam and your answers from your own computer.

How to take the exam Before starting, make sure that you are in a quiet place where you can work without interruption. To avoid distraction, please turn off your phone before starting the exam. You will not need any materials beside your computer and a reliable internet connection. Make sure your computer is sufficiently charged or plugged in to a power source. In addition you may want to have some scrap paper and a pen or pencil.

Download the file **exam2.txt** and open it in your text editor. Enter your name and NYU NetID at the beginning of the file. At the section "**Affirmation**" enter the following text exactly as it appears, substituting your name. Without a typed affirmation, your exam will not be graded. **Do this before you answer any questions in the exam!**

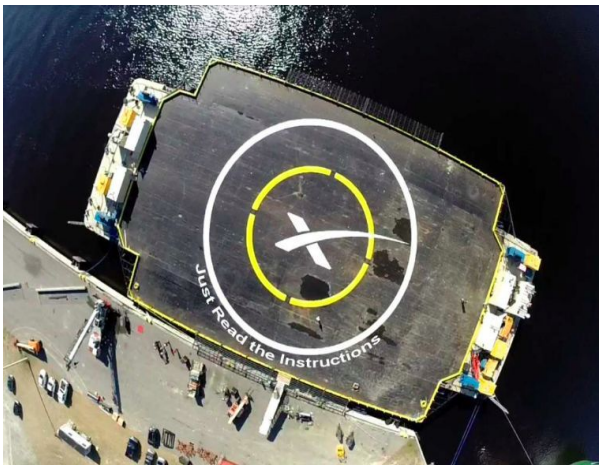
> I, *[your name]*, affirm that I have completed the exam completely on my own without consulting outside resources. I have followed the required rules. I understand that violating any of these rules would represent academic dishonesty.

Enter all of your exam answers into exam.txt at the appropriate places. You will have **80 minutes** to complete the exam. At the end of that period, you will have a **five minute** window to upload your completed exam to NYU Classes. After that time no work will be accepted.

You may upload as many times as you like. The last one before the deadline is what will be graded.

CS2124 Exam Two
2020 Spring

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. *Please* don’t assume any question you have seen before is the same as what you are being asked!

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.

Questions	Points
1	xtra
2-11	5
12	50

If you think you have seen one of these questions before, there is a good chance you are mistaken.

Best of luck!

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

a) Gosling
b) Hopper
c) Ritchie
d) Stroustrup

e) Thompson
f) van Rosum
g) Wall
h) None of the above

2. Given:

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

int main() {
    int y = 42;
    Thing(y);
    cout << y << endl;
}
```

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 42
i) The program will print out: 42 17
j) The program will print out: 42 28
k) The program will print out: 42 42
l) All of the above
m) None of the above.

3. Given:

```
class Parent {
public:
    virtual void Thing() = 0;                       // Line A
};

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

class GrandChild : public Child { };

int main() {
    GrandChild gc;                                  // Line C
    Parent* pp = &gc;                               // Line D
    pp->Thing();                                    // Line E
    gc.Thing();                                     // Line F

}
```

What will happen when we build and run the program?

a) It will fail to compile at line A

b) It will fail to compile at line B

c) It will fail to compile at line C

d) It will fail to compile at lines D and E

e) It will fail to compile at line F

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

g) It will print out:
Child
Child

i) It will print out:
Child
GrandChild

j) It will print out:
GrandChild
Child

k) It will print out:
GrandChild
GrandChild

l) None of the above

4. Given:

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

int main() {
    Thing thingOne("abc", 0);
    string s = "def";
    thingOne = s;
    thingOne.display();
}
```

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.

5. Given:

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

class Derived : public Base {
public:
    void derMeth() {
        protectedMethod();    // line A
    }
};

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

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
e. All of the above
f. None of the above

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() {
    Pet* petP = new Cat();
    Cat* catP = petP;
    catP->eat();
}
```

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.

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& argA, Base& argB) {
    argA.method(argB);
}

int main() {
    Derived d;
    Base b;
    someFunc(d, b);
}
```

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

8. Given

```
int* data = new int[12];
```

Which of the following is equivalent to **`data[5]`**?
(Note that there is ONLY ONE correct answer.)

a) data[5]	d) *data+5	g) &(data+5)	j) data+5&
b) &(data*5)	e) *(data+5)	h) (data+5)&	k) data&+5
c) &data+5	f) (data+5)*	i) data+5	l) data*5

9. Given:

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

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

what would be the result of:

```
int main() {
     HangGlider hanger;
     FlyingMachine flier;
     flier = hanger;
     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

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";
    }
};
```

what would be the result of:

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

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

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;
}
```

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

Answer question 12 in your blue book.

12. Define a class **BroodWar**

The class BroodWar will inherit from the class Starcraft.

Starcraft
- has a constructor that takes an int 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 Starcraft.
- NB: you are **not responsible** for defining the Starcraft class.

BroodWar has two fields, the player's name and a collection of Zergling pointers. There *may be* lots of different types of Zerglings, but we won't be responsible for defining those derived classes here.

The Zerglings will all be on the heap. Do not worry about how they got there. They just are.

You are **not responsible** for defining the Zergling class.
Zerglings support copy control, along with all necessary operators

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

A constructor taking in the player's name and registration code.

Copy **control**. Yes, **all** of it.
- 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 BroodWar instance should be included.
- Don't worry about printing information contained in the Starcraft class.

An equality operator.
- Two BroodWar instances are considered equal if all of the *corresponding* Zerglings are equal, i.e.:
  - there are the same number of Zerglings
  - each Zergling in one BroodWar matches (is equal to) the Zergling in the same position of the collection in the other BroodWar.
- NB, the Zerglings do not have to have the same address in memory to be "equal".

Given that `brood` is an instance of `BroodWar`

Then the code:

```
if (brood) { cout << "There be Zerglings!"; }
```

should print out "There be Zerglings!" if and only if that brood has any zerglings.