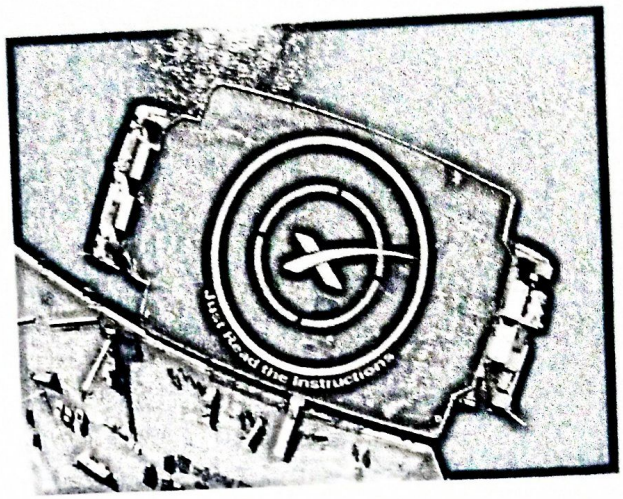
Name: Ama Khusheed NetID amk960

CS2124 Exam Two

2018 Fall



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.

Questions	Points
1	xtra
2-9	5
10	10
11	50

Answer questions 1–10 in the exam book. For multiple choice questions, **circle** the correct answer. There should be only one correct answer / question.

Answer questions 11 in your blue book.

Place your name and id on every page in this book before the end of the exam. You will lose points if you have not done so when we call "time to put your pens / pencils down." Sorry to be punitive, but some students seem to want to get that small advantage over their classmates.

Name: Asma Khursheed NetID amk960

For multiple choice questions CIRCLE the right answer

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 foo(int x) {
    const int* p = &x;    // line A
    x = 28;               // line B
    cout << *p << ' ';    // line C
    *p = 42;              // line D
}

int main() {
    int y = 17;
    foo(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 runtime 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: 28 17
j) The program will print out: 28 42
k) The program will print out: 42 17
l) All of the above
m) None of the above.

Name: Asma Khusheed NetID amk960

3. Given:

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

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

class GrandChild : public Child { };

int main() {                                        // Line B
    GrandChild gc;                                  // Line C
    gc.foo();
}
```

What will happen when we build and run the program?

a) It will fail to compile at line A, because **foo is marked virtual in Child**

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

c) It will fail to compile at line C because **foo is an abstract method.**

d) It will fail to compile for some other reason

e) It will print out:
`Parent`

f) It will print out:
`Child`

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

h) None of the above

4. Given:

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

Pick an expression that is equivalent to: **`&data[5]`**

data[5] = *(data+5)
&data[5] = data+5

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

Name: Asma Khurshood NetID amk9460

5. Given:

```
class Pet {
public:
    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();
}
```

derived cannot point to base.

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

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

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

c. The program fails to compile because the method eat is not marked virtual.

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

e. The program compiles and crashes when it runs.

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

g. None of the above.

Name: Asma Khurshеed NetID amk960

6. 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 der;
    someFunc(der, der);
}
```

derived derived

der. 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

Name: Asma Khursheed NetID amk960

7. Given

```
class Base {
public:
    virtual 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.

Name: Asma Khursheed NetID amk960

8. 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;
    flier.fly();
}
```

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

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

c. Runtime error.

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

e. Compilation error because fly is not virtual.

f. Other compilation error.

g. None of the above

Name: Asma Khurshеed NetID amk960

9. Given:

```
class Member {
public:
    Member()  {cout << 1;}
    ~Member() {cout << 2;}
};

class Base {
public:
    Base()  {cout << 3;}
    ~Base() {cout << 4;}
};

class Derived : public Base {
    Member member;
public:
    Derived()  {cout << 5;}
    ~Derived() {cout << 6;}
};

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

315624

What is the output?

a. 135246

b. 135642

c. 153264

d. 153462

e. 315426

f. 315624

g. 351462

h. 351264

i. 513624

j. 513426

k. 531642

l. 531246

m. Fails to compile

n. Runtime error (or undefined behavior)

o. None of the above

Name: Asma Khurshed NetID amk960

10. Given

- a class `Base` and
- a class `Derived`
  - that publicly inherits from `Base`
  - and has an `int` member variable named `foo`
  - nothing else

a. Write a copy constructor for `Derived`.

```
Derived(const Derived& rhs): Base(), foo(rhs.foo){}
```

-3

b. Write an assignment operator for `Derived`.

```
Derived& operator=(const Derived& rhs) {
    Base::operator=(rhs);
    foo = rhs.foo;
    return *this;
}
```

c. Given the code

```
Base* bp = new Derived();
delete bp;
```

What is necessary for the `Derived` destructor be called?

To make the destructor for Base class, as virtual. This ensures it waits till runtime, calls the Derived destructor.

Name: Asma Khurshеed NetID amk960

Answer question 12 in your blue book.

11. You will define / implement the three classes `Company`, `Employee` and `Geek` described below

- Your code should generate the same output as shown on the next page
- Companies hire employees.
  - One company can have many employees.
  - One employee can belong to at most one company.
- There are many types of employees: managers, secretaries, security, building maintenance and, of course, geeks!
  - The only employee classes that you have to implement are `Employee` and `Geek`. Other programmers will implement the other types of employees.
  - Employees all have names and salaries.
  - The different employee types are each special in some way.
    - Geeks, for example, have something called *Nerd Power*.
      - It is composed of their IQ and the number of baths that they take in a year.
      - *Assume* there is a class `NerdPower`. But someone else is writing it.
      - The class `Geek` will need to have a `NerdPower` *member variable*.
      - `NerdPower` will have a constructor that takes the two integer values representing the IQ and the number of baths taken per year.
      - `NerdPower` will have an overloaded output operator.
    - Managers are known for how lame they are.
      But, again, you are not implementing the `Manager` class, so you don't really need to know that.
      You may assume the Manager class has any functions overloaded that are needed for the test code to work.
  - **Hiring**: a company is *not* allowed to hire someone who already has a job.
    - Happily we only have to worry about hiring them. Our recruiting department is so good that we never need to fire anyone and everyone is so happy that they don't want to quit!
- Displaying:
  - each *type* of employee will have different stuff to be displayed ~~in addition to their name and salary~~.
    Your code for `Company` should still work even if new types of employees are created.
  - Note that we use an **output operator** to display the Company
  - The company in turn uses output operator(s) to display its employees.
  - Each new type of Employee is ***responsible*** for correctly displaying itself.
    - And no, we never create an instance of the Employee class.
- **Note that you can ask an employee how many people there are in the company he or she is a member of.**
  There is an example in the sample where Dilbert provides that information (for the Losers R Us company).
- **Do NOT copy the test code into your blue book unless you really just like wasting time.**

Name: Asma Khurshed NetID amk960

Test Code

```
int main() {

    Company losers("Losers R Us");

    // Dilbert, a Geek, has a salary of $200K,
    // has IQ of 150 and takes 20 baths per year.
    Geek dilbert("Dilbert", 200, 150, 20);

    // Wally's, another Geek, but has a salary of only $150K,
    // an IQ of 150 and takes 10 baths per year.
    Geek wally("Wally", 150, 120, 10);

    // Pointy Hair is a manager, has a salary
    // of $50K, and has a lameness value of 1000.
    Manager pointy("Pointy Hair", 50, 1000);

    loser.hire(dilbert);
    loser.hire(wally);
    loser.hire(pointy);

    cout << "================\n";
    cout << loser;
    cout << "================\n";
    cout << "Dilbert says the company's size is: "
         << dilbert.getDivSize() << endl;
}
```

Output for Test Code

```
================
Company: Losers R Us
Geek: Dilbert; Salary: $200K; IQ: 150; Baths per year: 20
Geek: Wally; Salary: $150K; IQ: 120; Baths per year: 10
Manager: Pointy Hair; Salary: $50K; Lameness: 1000
================
Dilbert says the company's size is: 3
```