[redacted]

# CS2124 Exam One

## 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, require that you read and *understand* the programs shown. You need to read them *carefully* if you are going to understand them.

If a question asks you to write a class or a function and shows you test code, be sure your class / function works with that test code.

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 | 4 |
| 2-6 | 5 |
| 7-9 | 9 |
| 10 | 16 |
| 11 | 32 |

Answer questions 1–9 in **this exam book**. For multiple choice questions, circle the correct answer. There should be only one correct answer / question.

Answer questions 10 and 11 in your **blue book**.

Place you name and id on every page in this book <u>before</u> the end of the exam.

[redacted]

For mulitple choice questions, circle one 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. ✓ The expression `*p.x` means the same thing as:

   a) `p->x`

   [✓ b)] `*(p.x)`

   c) all of the above

   d) none of the above

3. Given a class called `Thing` and the code

```
Thing thingOne;
```

What function **call** is the following line equivalent to?

```
Thing thingTwo = thingOne;
```

✓

   a) `Thing& Thing::operator=(const Thing& rhs)`

   b) `operator=(thingTwo, thingOne)`

   c) `thingOne.operator=(thingTwo)`

   d) Either (b) or (c), depending on how the programmer chose to implement the operator.

   [✓ e)] None of the above because it is using the `Thing` copy constructor.

   f) None of the above

[redacted]

4. Given:

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

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

constant pointer to int

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]

answer ① [✓ 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 42

h) The program will print out: 42 17

i) The program will print out: 42 42

j) The program will print out: 17 28

k) The program will print out: 42 28

l) All of the above

m) None of the above.

Name: [redacted]

5. Given the following code:

```
class Dragon {          defaulti val = Nadstrer
public:
    Dragon(string val = "Roadster") : payload(val) {}
    void display() { cout << "Dragon payload: " << payload << endl; }
private:
    string payload;
}; str val= "Roadster"

class Falcon {
public:
    void display() {
        cout << "Falcon class ship\n";
        fly.display();
    }
private:
    Dragon fly; ~~&#~~ Ovrloaded default constuctor    Dragon fly("Roaster")
};

int main() {
    Falcon heavy;             Falcon class ship
    heavy.display();          Dragon paylod : Roadster
}
```

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

✓ a) Outputs:
```
Dragon payload: Roadster
```

b) Outputs:
```
Falcon class ship
```

c) Outputs:
```
Dragon payload: Roadster
Falcon class ship
```

[✓ d)] Outputs:
```
Falcon class ship
Dragon payload: Roadster
```

~~e)~~ Compile time error

f) Run time error (or undefined behavior)

g) None of the above.

6. Given a vector of strings, called `strVec`, use a ranged for (also known as the "foreach"), to print the items in the vector to the standard output, one per line.
(No, do not put this in a function.)

```
     const       &
for (string str : strVec) {
    cout<< str << endl;
}
```

1

Name: [redacted]

7. Given:

```
class Thing {
public:
    void display() { cout << name; }
    string name;
};

int main() {
    Thing x;
    string fred = "fred";

    x = fred;

    x.display();
}
```

Modify the Thing class by adding public method(s) so that main will result in displaying the string “fred”. Do not modify display or main. Just add a new method to Thing (or methods, if you like.). Write the method(s) below. Before you ask, no you don’t have to qualify them with Thing::.

~~void~~ ~~Thing& operator=(const string& name)~~ rhsName {

~~void~~ ~~name=~~

```
Thing& operator=(const ~~Thing&~~ string& rhsName) {
    ~~Thing newThing;~~
    ~~newThing~~.name = ~~rhs.name~~ rhsName;
    return ~~newThing~~;
}
```

3

Name [redacted]

8. Given:

```
class ShuttleCraft {
public:
    ShuttleCraft(string s) : s(s) {}
    // ... possibly other methods
private:
    string s;
    // ... possibly other fields
};

class Orville {
public:
    Orville(string commander, string s)
        : commander(commander), p(new ShuttleCraft(s)) {}
    ~Orville() { delete p; }
private:
    ShuttleCraft* p;
    string commander;
};
```

Implement an appropriate assignment operator, i.e. a deep copy, for the class `Orville`. Write it below. (Oh, yes, the class `ShuttleCraft` supports copy control.)

~~ShuttleCraft& operator=(const ShuttleCraft~~

```
Orville& operator=(const Orville& rhs) {
        if (&rhs != this) {
            delete p;
            p = new Shuttlecraft(*rhs.p);
            commander = ?
        }
        return *this;

}
```

8

[redacted]

pointer to int pointers

9. Given: → int** p =new int* [100]

~~int* dai = new int[]~~

a) Define a variable `dai` that points to a dynamic array of int pointers.

```
Int ** p = new int* [ ]
~~vector<int*>~~ dai
~~vector<int*> dai;~~
```

b) Fill the array with addresses of 100 ints that you will allocate on the heap. The ints will have values from 1 to 100.

```
for (size_t i=0; i < ~~dai.size()~~ 100; ++i ){
        ~~dai.push_back(new int(i));~~
        ~~dai->push_back(new int(i));~~
}
```

dai[i]= new int(i); i+1

c) Now, modify those values by adding the index of the entry to the *value* that was stored on the heap, e.g. add 17 to integer pointed to by `dai[17]`.

```
for (size_t i=0; i< 100; ++i) {
        *dai[i] += i
}
```

✓

d) Finally, free up all of the space you allocated on the heap.

```
for (size_t i=0; i< dai.size(); ++i) {
        delete dai[i];
        dai[i]=nullptr;
}
```

delete [ ] dai;

4

[redacted]

**Blue book**

**Answer questions 10 and 11 in your blue book.**

10. Given the type `Thing`:

```
struct Thing {
    vector<int> stuff;
};
```

write the following two functions,

**`fill`**: fills a vector of Things with data from a file stream.

- The lines of the file each
  - start with a string “one” or “three”.
  - And then have that many ints on the rest of the line.
  - Example file:
    three 2 4 6
    one 12
    one 18
- Put the ints that show up on a single line into the vector in a Thing object. There should be one Thing object for each line in the file.
- Note the stream is already open, so you don't have to worry about that. And we are closing it for you, so you don’t have to worry about that either.

**`totalStuff`**:

- Passed the vector of Things. Computes and returns a single `int` which is the total of all the ints in all of Things in the vector.
- For the above sample input file the function would return the sum 2+4+6+12+18 (so I think the answer is 42).

Below is an example program in which `fill` and `totalStuff` are called from main.

- Note that neither `fill` nor `totalStuff` are *methods*.
- Do not modify the `Thing` struct.

```
int main() {
    ifstream ifs("stuff.txt");
    vector<Thing> things;        // Note, not Thing pointers
    fill(ifs, things);           // Implement this function
    ifs.close();
    cout << “Total stuff: “
         << totalStuff(things)  // Implement this function
         << endl;
}
```

Answer in the blue book!!!

[redacted]

11. The latest big startup is Itsy. They connect people who are looking to hire other people or else to get hired. It's important for their business model to keep track of all the bits of business they get, so for every client (known as a Bit) they track who the Bit is currently working for, i.e his boss, and which Bits he has hired, his team. Everyone who hires and everyone who is hired is a Bit, so they all form a bit of a community.

    What to you need to do?

    - Define a class **`Bit`** to represent a member of the community.
        - `Bit`s do have names, by the way, but of course they are **not** unique.
    - Track who each `Bit` is currently employing. There can be <u>many</u> of them. <~~4~~ vector<Bit*>
    - Track who each `Bit` is currently working for. There can be <u>only one</u> (at a time)! Bit* boss
    - Allow a `Bit` to hire another `Bit` with a **`hire`** method.
    - Allow a `Bit` to quit with a **`quit`** method.
    - Enforce a few rules:
        - When hiring, you cannot hire someone who:
            - has a job
            - you work for
            - who works for you
            - and of course you can't hire yourself
        - If an attempt to hire fails, don't fail silently! (You know what that means.)
        - `Bit`s who employee other `Bit`s don't particularly care what order the `Bit`s in their team are kept in, so when one quits, it should only be an O(1), i.e. constant time, operation.

Note the output. Your **output operator** should generate the output *as shown*, except possibly for the order of the Bits in a group. (Remember, we don't care about order.)

And finally, no, this problem does **<u>not involve copy control or the heap</u>**.

And even more finally, do note that people's names are ***not*** unique!!! Just knowing someone's name won't be very useful.

**[Example test program and output on the next page]**

[redacted]

### Test Code:

```
int main() {
    Bit moe("Moe");
    Bit larry("Larry");
    Bit curly("Curly");
    Bit curly2("Curly");

    larry.hire(moe);    // Returns true
    cout << larry << endl;
    larry.hire(curly);  // Returns true
    larry.hire(curly2); // Returns true. Now we have two chips named
                        // Curly in the team
    cout << larry << endl;
    moe.hire(larry);    // Returns false.  Can't hire own boss
    moe.hire(moe);      // Returns false.  Can't hire self
    moe.quit();
    cout << moe << endl;
    moe.hire(larry);    // Returns true.
    cout << moe << endl;
    curly2.hire(moe);   // Returns true.  We are allowed to have a cycle
    cout << moe << endl;
    cout << larry << endl;
    cout << curly2 << endl;
}
```

### Output:

```
Name: Larry; Boss: none; Bits: Moe.
Name: Larry; Boss: none; Bits: Moe Curly Curly.
Name: Moe; Boss: none; Bits: none.
Name: Moe; Boss: none; Bits: Larry.
Name: Moe; Boss: Curly; Bits: Larry.
Name: Larry; Boss: Moe; Bits: Curly Curly.
Name: Curly; Boss: Larry; Bits: Moe.
```

Write your answer in the Blue Book!