

433 Practice Midterm

1. (15 points - virtual method invocation) What is printed by this program?

```
#include <stream.h>

class A {
public:
    virtual void f() { cout << "A::f()" << endl; }
    void g() { cout << "A::g()" << endl; }
    void h() { cout << "A::h()" << endl;
              f();
              g();
            }
};

class B : public A {
public:
    void f() { cout << "B::f()" << endl; }
    void g() { cout << "B::g()" << endl; }
};

void main(int argc, char ** argv) {
    A a;
    a.h();
    B b;
    b.h();
    A * p = new B();
    p->h();
    delete p;
}
```

2. (15 points - Parameter passing) For each of the following functions, describe the effects of invoking the function on the two Point values passed as parameters to the function. Assume they are passed appropriately (i.e., there are no compilation errors).

```
// C++
void f(Point p, Point q) {
    p.x = 42;
    p = q;
}
// C++
void h(Point * p, Point * q) {
    p->x = 42;
    p = q;
}

// C++
void g(Point & p, Point & q) {
    p.x = 42;
    p = q;
}
// Java
void i(Point p, Point q) {
    p.x = 42;
    p = q;
}
```

3. (15 points) Say you have class A, and class B being a subtype of A. In C++, how are arrays of A and arrays of B handled? In particular, can you pass one to a function that expects the other? Describe how it works and any issues or problems.

What about Java?

Back to C++: what if you had an array of pointers to A and an array of pointers to B. Should this work? (I don't expect you to have the C++ standard memorized, so I don't expect you to tell me what the standard says)? Why or why not? What are the issues?

4. (25 points – C++ design) Consider the skeleton classes given below. Define appropriate “hidden” functions for these classes (e.g., copy constructor, ...). Assume that no void constructor should exist for either class. Don’t prohibit anything that might be reasonable, and guard against memory leaks. If you have a choice about whether to allow an operation or what semantics to provide, describe your choices and then chose one. You should not change the data structures to something “better”. Assume that reasonable hidden functions are defined for `Point` and `Color`. As much as possible, you should try to make it so that changes to `DisplayedText` only mean that `DisplayedColoredText` needs to be recompiled, not rewritten; if changes are required to `DisplayedColoredText`, they should be as small as possible.

```
class DisplayedText {
public:
    DisplayedText(char * buf, Point p) : pos(p) {
        txt = new char[1+strlen(buf)];
        strcpy(txt,buf);
    }
private:
    char * txt; // ref to malloc'd, null-terminated string
    Point pos; // position text is to be displayed at
};

class DisplayedColorText : public DisplayedText {
public:
    DisplayedColorText(char *buf, Point p, Color c) : DisplayedText(buf, p),
                                                    clr(c) {};
private:
    Color clr; // Color of text
};
```

5. (15 points) What are the possible results of this code fragment? What is the effect of the synchronization?

```
Point a = new Point(1,2);
Point b = new Point(3,4);

Thread t1 = new Thread() {
    public void run() {
        synchronized(a) {
            a.x = b.x;
            a.y = b.y;
        }
    }
};

Thread t2 = new Thread() {
    public void run() {
        synchronized(b) {
            b.y = 40;
            b.x = 30;
        }
    }
};

t1.start(); t2.start();
t1.join(); t2.join();
System.out.println("a = " + a);
System.out.println("b = " + b);
```

6. (15 points) Consider the following Java function:

```
static int f(int i, int j) {
    while (true) {
        try {
            if (i <= 0) throw new IllegalArgumentException();
            if (i == 1) return 42;
            i--;
        }
        finally {
            if (j == 0) {
                j++;
                continue;
            }
            if (j < 0) throw new IllegalStateException();
            if (j > 10) return 17;
        }
    }
}
```

What does this function do for each of the following sets of arguments?

| | | | |
|-----------|----------|----------|-----------|
| f(-1, -1) | f(-1, 0) | f(-1, 5) | f(-1, 20) |
| f(1, -1) | f(1, 0) | f(1, 5) | f(1, 20) |