

---

**Midterm (30 pts)**

## 1 Conditional (4 pts)

Java's boolean type enhances reliability by enabling automatic detection of mistakes such as using "=" for "==" in a fragment such as — "int i; if (i = 0) {}";. Can you conjure up a scenerio in Java where this typo cannot be detected by the type system?

## 2 Arrays (8 pts)

Draw "box-arrow-cloud" diagram approximating run-time data structures created after *all* the variable definitions (formals and locals) and value assignments are executed in the following Java code:

```
public class Exam1 {
    public static void main(String [] args) {
        x = new Exam1();
        Exam1[] y = {x, null, x, new Exam1()};
    }
}

% javac Exam1.java
% java Exam1 0 abc
```

Typically, an object is labelled with the name of the class it is an instance of and its state (the values contained in the various fields).

## 3 Operators (6 pts)

Arithmetic subtraction is not an associative operation, so binary minus (-) is not an associative operator. Arithmetic addition and string concatenation are both associative operations, but Java regards binary plus (+) as a left-associative operator, to disambiguate string expressions involving both operations. Arithmetic multiplication is an associative operation. Do you see any reason to specify associativity for the \* operator in Java? Explain.

## 4 Dynamic Binding (4 + 2 + 6 pts)

Does the following program compile without errors? If so, proceed further. If not, change the program "minimally" to obtain an error-free compile. (The permitted changes are: (i) addition of a suitable cast, or (ii) deletion of a statement that cannot be corrected using a cast.)

```
class P {
    public void f(C c) {
        System.out.println("f(C) in P. ");
    }
}
class C extends P {
    public void f(P p) {
        System.out.println("f(P) in C. ");
    }
    public void f(C p) {
        System.out.println("f(C) in C. ");
    }
}
class Exam2 {
    public static void main(String[] args) {
        P pp = new P();
        P pc = new C();
        C cc = pc;

        pp.f((C) pc);
        pc.f(pp);
        pc.f(cc);
        cc.f(pc);
        cc.f(cc);
        ((P) cc).f(cc);
    }
}
```

Does the modified program run without exceptions? If so, what is the printed output? If not, delete all the objectionable statements, and then determine the printed output.