CS164: Written Assignment 6
(On Implementation of Object-Oriented Languages)

Assigned: Thursday, Oct 21, 2004
Due: Tuesday, Oct 28, 2004, at the beginning of class.

Grading and Submission

Your answers must be brief and easy to understand. Your grade (credit/no credit) will depend partly on how easy it is for us to understand and verify your answer. If the written assignment contains multiple questions or multiple parts, **you must answer all questions to receive a "pass" grade.** Minor mistakes are acceptable. Submit your written assignments either in the classroom (before the lecture) or in 283 Soda. *No late homeworks are accepted.* **Please indicate your login name and Section number.**

1 Compiling Object-Oriented Programs.

Consider the following Java program:

```
class A {
   int a = 0;
   int x() { return a; }
};
class B extends A {
   int b = 2;
   int e = 4;
   int y() { a = a - b + e; return a; }
};
class C extends A {
   int c = 3;
   int e = 5;
   int x() { return a + e; }
   int z() { e = a * c; return a; }
};
class Test {
   public static void main (String[] args) {
      A p = new B();
      B r = new B();
      C s = new C();
      A t = new A();
      p.x();  r.x();  s.x();  t.x();
   }
};
```

Answer the following questions:

1. The program instantiates three classes of objects (A, B, and C). Show the object layout for each class. Show both the programmer-accessible fields and the "header" fields.

2. Can the e fields in classes B and C be placed at different offsets? **Clearly and concisely** justify your answer (your answer should be to the point; don't put down any arguments irrelevant to the question).

3. How many different methods (i.e., assembly-level procedures) will be generated by the compiler for the above program? What are the names of these methods? Give their names in the "assemblerized" form `_classname_methodname`.

4. Draw the memory content at the end of *main*. Show the pointer links between the pointer variables (p, r, s, t), objects, dispatch tables, and procedures.

   Note: Your picture should have four kinds of nodes (pointer variables, objects, dispatch tables, and procedures) and one kind of edge (denoting points-to relationship). The content of each pointer in the picture should be depicted as an edge to the target of the pointer.

5. Give the sequence of x86 instructions that implements the dynamic dispatch call $t.x()$.

   Assume that the value of the variable t is stored in register %eax. Assume that the dispatch table and the object header are laid out as you defined above. Comment your code fragment: what does each offset mean? what does each register contain?

6. The *dynamic-dispatch* call used to implement non-static method calls in object-oriented languages is more expensive than the *direct* procedure call that you used to implement invocations of static methods in PA4. Fortunately, in some cases, the compiler can remove the dynamic dispatch and replace it with a direct procedure call.

   Question: in which of the four calls in the above program can the dynamic dispatch be replaced with a direct call? (Assume that the compiler does not "see" the *new* statements; i.e., it only knows the type of p, not the type of the object pointed to by p)? Clearly and concisely explain why your optimization is possible.

   (You can also assume that the program is guaranteed not to load any subclasses of B or C at run-time.)