Spring, 1996 Name: ______________________

Com S 541 — Programming Languages 1

Final Exam on OO, Smalltalk, and Parameterization

Special Directions for this Test

This test has 12 questions and pages numbered 1 through 7.
You are allowed to use a copy of TR91-22a and two pages of notes. These notes are to be handed in at the end of the test.
Before you begin, please take a moment to look over the entire test so that you can budget your time.
In Smalltalk programs, you can use anything that is documented in TR91-22a. **If you want anything else, you must write it into your answer.**

For each of the following multiple choice questions: (i) circle the answer that best completes the sentence, and (ii) write one or two sentences to briefly justify your answer.

1. (10 points) [[This turned out to be somewhat ambiguous]] A Smalltalk object is most like the following from Schmidt's book:
   (a) a class.
   (b) a type structure.
   (c) a record.
   (d) a variable.
   (e) a location.
   (f) a module.

2. (10 points) In Smalltalk, an abstract class is a class that:
   (a) has a lot of mathematics in it.
   (b) summarizes journal papers.
   (c) has more than one subclasses.
   (d) has no subclasses, or only one subclass.
   (e) should not be instantiated.
   (f) should be instantiated.

3. (10 points) The difference between a subclass, a subtype, and a behavioral subtype is that:
   (a) A subclass inherits implementation from its superclasses, a subtype has the same protocol as its supertypes, and a behavioral subtype has the same behavioral protocol as its behavioral supertypes.
   (b) A subclass inherits protocol from its superclasses, a subtype has the same implementation as its supertypes, and a behavioral subtype has the same behavioral implementation as its behavioral supertypes.
   (c) A subclass inherits data from its superclasses, a subtype has the same methods as its supertypes, and a behavioral subtype inherits both data and methods from its behavioral supertypes.
   (d) A subclass is a concrete class, a subtype is a concrete class that inherits from an abstract class, and a behavioral subtype is an abstract class that inherits from an abstract class.
   (e) There is no difference, all of these concepts are the same.
   (f) There is no difference between subtype and subclass, but a behavioral subtype is something that Kishore works on.

For each of the following multiple choice questions: (i) circle the answer that best completes the sentence, and (ii) write one or two sentences to briefly justify your answer.

4. (10 points) [[This turned out to be somewhat ambiguous]] Smalltalk:

   (a) mostly satisfies the parameterization principle, because methods can take parameters, methods are expression abstractions, and expressions are the main semantically meaningful syntactic category.

   (b) mostly satisfies the parameterization principle, because expressions can be parameters, and expressions are the main semantically meaningful syntactic category.

   (c) mostly satisfies the parameterization principle, because blocks in Smalltalk are like lambda abstractions.

   (d) does not satisfy the parameterization principle, because classes do not take parameters.

5. (10 points) The programming language SML uses structural-equivalence typing for its record types. Does it make sense for SML to allow a record type to be defined the a formal parameter list, as in the following?

```
fun question(pnt : {x: int, y: int}) = ...
```

   (a) yes.

   (b) no.

   (c) it depends on the compiler.

   (d) there is not enough information in the problem to tell.

6. (10 points) Give an example of a syntactic domain (from Schmidt's imperative language) where the abstraction form should be evaluated one way (for example, lazily), and the corresponding parameter form is best evaluated in another way (for example, eagerly).

7. (10 points) List the (a) advantages and (b) disadvantages of dynamic type checking.

8. (15 points) Give an equivalent name for each of the following parameter mechanisms.

   (a) eagerly-evaluated expression parameters

   (b) lazily-evaluated expression parameters

   (c) eagerly-evaluated variable parameters

9. (20 points) (a) Give an example of a program (in Schmidt's language) in which call-by-value-result and call-by-reference give different results. (Hint: you can use parameter lists if you wish.) (b) For both of these parameter mechanisms, give the output of the program (i.e., the final state of the store).

10. (20 points) Suppose that class `C1` is a subclass of class `Object`, and that `C2` is a subclass of `C1`. The methods of each class follow the appropriate comment and are separated by blank lines.

```
"methods of class C1"
larry: x
   ^ 2 * x

moe: x
   ^ 5 + x

curly: x
   ^ self sue: x

sue: x
   ^ self gayle: x

gayle: x
   ^ 5 + x


"methods of class C2"
larry: x
   ^ super moe: x

moe: x
   ^ self curly: x

gayle: x
   ^ 7 + x
```

For each of the following expressions, describe the execution (in words, especially noting which methods of which classes are called), and give the result of the expression (if any). If the expression does not terminate, say that and explain why.

(a) `(C1 new) curly: 8`

(b) `(C2 new) moe: 9`

11. (20 points) [[This was too hard, and needs better examples]] In Smalltalk, write a method, `isSublistOf:` for class `List` such that

```
lst1 isSublistOf: lst2
```

is `true` just when the elements of `lst1`, occur consecutively in `lst2`. For example, consider the following transcript.

```
>    lst12 <- ((List new) addLast: 1) addLast: 2
List ( 1 2 )
>    lst162 <- (((List new) addLast: 1) addLast: 6) addLast: 2
List ( 1 6 2 )
>    lst5126 <- ((((List new) addLast: 5) addLast: 1) addLast: 2) addLast: 6
List ( 5 1 2 6 )
>    lst12 isSublistOf: list5126
true
>    (List new) isSublistOf: lst5126
true
>    lst5126 isSublistOf: lst12
false
>    lst162 isSublistOf: lst5126
false
>    lst12 isSublistOf: lst162
false
```

The method `isSublistOf:` should not modify any of its arguments (If you need to, make a copy and then modify the copy.) Use `==` to compare elements of the lists.

12. In this problem you will design and implement a class `BinTree` of binary trees as a subclass of `Collection`. This can either be done as a single concrete class, or as a small hierarchy of classes. You should think about the design a bit, then answer the following questions.

    (a) (5 points) What would you type to the Little Smalltalk system prompt to define the class `BinTree` (and any other classes you may need)?

    (b) (30 points) The class `BinTree` should have at least eight (8) methods that it does not inherit from its superclass. These methods should include the `size` and `do:` methods so that a `BinTree` can act like a `Collection` (see sections 6.1.1 and 6.1.2 of TR91-22a) and a `new` method.

    Using the convention that a completely blank line separates methods, write code for the methods below. If you have more than one class, indicate with comment lines which methods belong in which classes.