May 18, 1990 **Name:** ____________________________

CS 342 — Principles of Programming Languages

Final Exam

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Answers to essay questions will be judged on their completeness and clarity. However, please do not feel obligated to fill up all the empty space on the page. Conversely, do not feel limited by the space on a page, since you may continue an answer on the back of a page if you wish.

For a multiple choice question, circle the letter of the best answer; there will only be one best answer, so do not circle more than one.

problem	subject	points	score
1	principles	15	
2	static vs. dynamic type checking	10	
3	Smalltalk's blocks	10	
4	inheritance	10	
5	Scheme	11	
6	static vs. dynamic links	6	
7	draw the run-time stack	13	
8	design	25	
Total		100	

This exam (only) counts for 30% of your final grade.

1. (15 points) For each of the following language features, (i) name one (1) principle of language design that the feature illustrates or violates, (ii) indicate whether the feature illustrates or violates the principle you have named, and (iii) give a one-sentence explanation of why or how the feature illustrates or violates the principle. Write your answers in the following form:

   ```
   Violates zero-one-infinity, because only 6 character names are allowed.
   ```

   (a) CLU's type-checking

   (b) CLU's clusters

   (c) CLU's exception handling mechanism

   (d) Smalltalk's inheritance mechanism

   (e) Scheme's lambda

2. (10 points) Which one of the following most accurately characterizes the differences between static and dynamic type checking. (Hint: choose one of the ones at the bottom).

   (a) CLU uses static type checking.

   (b) Scheme uses static type checking.

   (c) Static type checking finds potential program errors earlier than dynamic type checking.

   (d) Dynamic type checking is more secure than static type checking, because some errors (namely type checking errors) are totally eliminated.

   (e) Dynamic type checking is necessary if a language wants to allow one to write a single (polymorphic) procedure that can sort arrays of reals and arrays of strings.

   (f) Static type checking is more efficient at run-time than dynamic type checking.

   (g) Static type checking is necessary for information hiding.

   (h) all of the above.

   (i) none of the above.

   (j) (a), (b), (f) and (g).

   (k) (a), (c), and (f).

   (l) (a), (c), (e), (f) and (g).

   (m) (b), (d), (e), and (g).

3. (10 points) Which of the following best characterizes Smalltalk's blocks.
   (a) Blocks are not needed in Smalltalk, except to write certain fancy programs.
   (b) Blocks allow Smalltalk to be very simple and regular.
   (c) Blocks are more powerful than Scheme's lambda abstractions.
   (d) Blocks are completely different than Scheme's lambda abstractions, so (c) is meaningless.
   (e) Programmers can used blocks to implement their own control structures in Smalltalk.
   (f) All of the above.
   (h) None of the above.
   (i) (c) and (e).
   (j) (a), (b), and (d).
   (k) (b) and (e).

4. (10 points) Which of the following best characterizes Smalltalk's inheritance (subclassing)mechanism?
   (a) The syntax used to create classes makes inheritance completely useless.
   (b) Inheritance promotes abstraction, because it allows one to reuse code from superclasses.
   (c) Inheritance violates abstraction, because a class can only have a single superclass.
   (d) Sending a message to the pseudo-variable `super` allows one to access the method of a superclass without dynamic binding.
   (e) Sending a message to `self` always invokes the method with the message's name defined in the class where the message is executed, if there is one.
   (f) All of the above.
   (h) None of the above.
   (i) (b), (c) and (d).
   (j) only (b) and (c).
   (k) (b), (d), and (e).
   (l) (a) and (e).

5. (11 points) Consider the following Scheme function definitions.

```
(define (test x)
  (lambda (y) x))

(define (problem z)
  ((lambda (x) ((test 'true) z))
   'blue))
```

What would Scheme return as the result, if any, for the expression `(problem 'apple)`?

6. (6 points) (a) Describe *where* the static link of an activation record points and the *purpose* (i.e., use) of the static link (in an implementation of a block-structured language that uses static chaining). (b) Also describe where the dynamic link points and its purpose.

7. (13 points) Consider the following Scheme function.

```
(define (problem a b)

    (define (foo x y)
        (define (bar z)
            ;;* draw the stack at this point
        )
      (bar x) )

    (define (snafu q s t)
       (foo (+ q s) t) )

  (snafu 3 a b) )
```

Draw the run-time stack, and any auxillary registers, for the execution of the expression `(problem 4 5)` when execution reaches the comment in `bar` using static chaining. (Please ignore Scheme's tail-recursion optimization.)

8. (25 points) Everyone on Route 128 is talking about B. Bittwiddle's new start-up Company "Trigonometry Inc." You are fortunate enough to be hired (on Professor Leavens's recommendation) as chief user-interface designer.

   Your task is to design the user-interface for version 1.0 of a computer program, `trigMaster`, that will allow surveyors, physicists, and high-school students to:

   - input trigonometric formulas (such as $z = \cos(x) + (\sin(y))^2$),
   - give values to the various variables in a formula, and calculate the result.

   B. Bittwiddle wants to get version 1.0 of `trigMaster` out the door fast, so your design should be functional, yet very simple. It has already been decided that version 1.0 of `trigMaster` will have a very simple, keyboard-style interface. (The graphical interface will be in version 2.0.)

   Design and describe the user-interface. In your description give an EBNF syntax and a brief English description of the meaning of each production (i.e., of each command in the grammar). Assume that the nonterminals ⟨identifier⟩ and ⟨number⟩ have already been defined, so don't describe their syntax; also don't worry if your grammar is ambiguous. Also list the names of any built-in (predefined) identifiers (and give their meanings if they are not obvious). (If you can't manage a grammar, at the very least give an English description of the operations and some examples of what the user would type.) It may help to think about what kinds of objects `trigMaster` manipulates and to work informally before writing a grammar.

(Answer for problem 8 continued)