CS 342 — Principles of Programming Languages March 6, 1991

Test

Please write your name here:

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 or answers. Each incorrect answer to a multiple choice question will result in the loss of one point.

problem	subject	points	score
1	design comparison	21	
2	scoping	5	
3	functional programming	19	
4	functionals	20	
5	interpreters	10	
6	first-class functions	10	
7	polymorphism	15	
Total		100	

This exam (only) counts for 20% of your final grade.

1. (21 points) Which of the following are correct statements about the design of Pascal and the programming languages we studied this term? You will lose 1 point for each incorrect answer, so if you aren't *sure* of the correct answer, you might not want to guess.

   (a) Pascal has first-class functions, like Scheme.

      i. true

      ii. false

   (b) The number of dimensions allowed for Pascal's arrays obeys the zero-one-infinity principle.

      i. true

      ii. false

   (c) Pascal gives programmers the same support for larger values found in APL, since Pascal has arrays.

      i. true

      ii. false

   (d) Pascal has as much automation as APL.

      i. true

      ii. false

   (e) Pascal allows one to write more modular and abstract programs than Scheme.

      i. true

      ii. false

   (f) Pascal is a simpler language than APL.

      i. true

      ii. false

   (g) Pascal allows one to write curried functions.

      i. true

      ii. false

2. Consider the following program:

```
(set s 7)
(set f (lambda (x) (* x s)))
(set g (lambda (s) (f (- s 11))))
```

   (a) (2 points) What is the value of `(g 12)` in a language with static scoping?

   (b) (3 points) What is the value of `(g 12)` in a language with dynamic scoping?

3. Consider the following LISP program:

```
(define foo (points)
  (begin
    (set points (* 100 points))
    (set count 55)
    (set res-list '())
    (while (< count 91)
       (begin
         (set temp (* count points))
         (set res-list (cons (/ temp 99) res-list))
         (set count (+ 5 count))))
    res-list))
```

(a) (9 points) Write a version of `foo` in LISP that does not use `set`, `begin`, or `while`. (That is, write the above without side-effects.) You may define auxiliary functions if you wish.

(b) (10 points) Write a version of `foo` in APL that does not use `while`, `if`, recursion, assignments that modify variables. Your function should return a vector whose elements are in the same order as the list returned by the version above. You may define auxiliary functions if you wish.

4. Consider the following Scheme program, which represents arrays whose elements are type α as functions of type int $\rightarrow \alpha$.

```
(set list->array
  (lambda (lst)
    (lambda (n)
      (nth lst n))))

(set array-fetch
  (lambda (arr) arr))

(set array-update
  (lambda (arr)
    (lambda (n)
      (lambda (x)
        (lambda (m)
          (if (= n m) x (arr m)))))))

(set nth (lambda (ls n) (if (= 0 n) (car ls) (nth (cdr ls) (- n 1)))))

(set mine (list->array '(2 3 7 27 703 10087)))
```

(a) (5 points) What is the value of `((array-fetch mine) 3)`?

(b) (5 points) Fill in the blanks in the following

`(set new-mine ______________ array-update mine ______________)`

so that after the above is executed, `((array-fetch new-mine) 3)` returns 100.

(c) (10 points) Suppose that one uses `array-update` to update the nth element of `mine` 50 times. How many calls of `=` are needed to fetch the $n+1$st element of `mine` after these updates?

5. (10 points) Describe the main differences between the Scheme and LISP interpreters. That is, how does the Scheme interpreter differ from the LISP interpreter. (Hint: if you don't know exactly, think about how the languages differ and then describe how these differences should be reflected in the interpreter.)

6. (10 points) What are the advantages of a language that has first-class functions (as in Scheme), versus one that does not? (Hint: you might compare Scheme to LISP)

7. (15 points) Compare the different approaches to polymorphism discussed in the Scheme chapter. Describe the advantages of each approach, and the trade-offs between the approaches. For what problems would each approach be best suited?