

Spring 2002 - Programming Languages Qualifier

Computer Science Department
University of Massachusetts Lowell
Lowell, MA 01854

Feb. 3, 2002.

Hand in problems 1-5 in one blue book; 6-12 in another. Do any 10.

1. Scheme.

Consider the following function f defined on the natural numbers:

- (i) $f(0, y) = y$
- (ii) $f(x+1, y) = 1+f(x, y)$

- (a) Write a Scheme function (procedure) `f1l` that computes f in linear time and space.
- (b) Write a Scheme function `f1c` that computes f in linear time and constant space.
- (c) Give a simple mathematical expression (closed form) for (the body of) f .
- (d) Prove that your answer to (c) is correct.
- (e) Write a Scheme function `fcc` that computes f in constant time and space.

2. Scheme.

- (a) What is the difference between ordinary procedures and special forms?
- (b) Describe advantages of being able to use ordinary procedures instead of special forms.
- (c) How could Scheme be modified to make dispensable many (but not all) of its special forms? List some commonly used special forms that could be discarded and illustrate how they could be implemented as ordinary procedures.

3. ML: Types.

Briefly describe type inference in ML. Be sure to indicate the difference between type checking and type inference. Give examples of types that admit equality. Give examples of types that do not admit equality and explain why they do not.

4. Logic Programming.

- (a) Describe the unification algorithm. Be sure to explicitly mark (underline) the occur check. Illustrate the results of applying the algorithm to some expressions.
- (b) Describe prenex conjunctive normal form. What is its significance?
- (c) Describe the resolution inference rule.

5. **Hoare Axiomatization.**

Assume that the assignment axiom is valid and the rules of inference are validity preserving. Prove the soundness of the Hoare axiomatization.

6. **ML: Curried Functions.**

Write a function `filter` that takes a predicate `P` as argument and produces a function that takes a list of elements of suitable type and returns those elements on the list that satisfy `P`.

7. **ML: Polymorphism.**

Given the functions

```
fun f(nil) = nil
  | f([x]) = [x]
  | f(x::y::zs) = [x, y];

fun g(x, y) = (f(x), f(y));

fun h(x, y) =
  let val v = f(nil) in (x::v, y::v) end;
```

For each of the expressions below, indicate whether it is legal, and, if not, what is the error?

- (a) `g([1,2,3], "a");`
- (b) `g([1,2,3], nil);`
- (c) `g([f,f], 1);`
- (d) `g([1], [1,0]);`
- (e) `h(1,2)`
- (f) `h(1, "a");`
- (g) `h(nil, nil);`
- (h) `h([1], nil);`

8. **Environments of Evaluation.**

Consider the following Scheme code in which one function is passed as an argument to another function. Note that the function called `pass-me` is variadic.

```
(define pass-me
  (lambda args
    (apply * args)))

(define calculate
  (lambda (proc a b c)
    (proc a b c)))
```

Using the diagramming notation of either Manis & Little or Abelson & Sussman, show the complete environment of evaluation during evaluation of

```
(calculate pass-me 2 3 4)
```

9. **Lambda Calculus.**

State the Church-Rosser Theorem and explain why it is important.

This will require you to explain what β -reduction is and its role in computation. I do not expect proofs.

10. **Exceptions.**

In ML (or Modula-3 or Ada, or any other language you know that provides exceptions), write a function `prod` to compute the product of a list of numbers. The function `prod` must return zero as soon as zero is encountered in the list **and not perform any unnecessary multiplications**.

11. **Parameter Passing.**

Consider the following function:

```
function F (x, y: integer) return integer is
begin
  x := x + 1;
  y := y + 1;
  return(x - y);
end F;
```

Show **by one or more examples of calls** on procedure `F` that call-by-name, call-by-value/result and call-by-reference are different parameter passing methods. That is, show calls that produce different results for the different binding rules.

12. **Denotational Semantics.**

- (a) What is meant by *Denotational Semantics*?
- (b) For the language of *Decimal Numerals* (numerals composed of decimal digits) describe:
 - The Syntactic Categories
 - The Syntax in BNF notation
 - The Value Domains
 - The Semantic Functions