

Spring 2004 - Programming Languages Qualifier

Computer Science Department
University of Massachusetts Lowell
Lowell, MA 01854

Feb. 7, 2004.

There are 10 problems on this exam. You should write the answers to problems 1–5 in one blue book and the answers to problems 6 – 10 in the other blue book.

1. Parameter passing mechanisms. (10 pts)

a) Consider the following program

```
int maybeAdd (int a, int b) {  
    int t;  
    while (b > 0) {  
        a = a + 1;  
        b = b - 1;  
    }  
    return a;  
}
```

```
main() {  
    int a = 2;  
    print (maybeAdd (a,a));  
    print a;  
}
```

What does this program print if the calling convention is

(2 pts) call-by-value?

(2 pts) call-by-reference?

(2 pts) call-by-value-return?

b) (4 pts) Write a short program in Scheme syntax that would give somehow behave differently in call-by-name (normal order) evaluation than in call-by-value (applicative order) evaluation, and explain the difference in the two behaviours for the program.

2. Exceptions. (10 pts)

a) (4 pts) Write a ML program to return the product of the elements in a list. The program should not perform any unnecessary multiplications if it finds a 0 in the list.

b) (4 pts) Briefly describe the exception mechanisms in two languages other than ML and compare them with each other and with ML's exception mechanism.

3. Concepts. (10 pts)

Briefly describe the similarities between the following concepts and their differences.

- a) (3 pts) l -values and r -values.
- b) (2 pts) partial correctness and total correctness.
- c) (2 pts) lists and sequences (aka streams).

d) (3 pts) Define Horn clauses and literals.

4. Fixpoints. (10 pts)

a) (2 pts) Describe Tarski's fixpoint construction.

b) (2 pts) Function $twom$ of type $(\mathbb{N} \rightarrow \mathbb{N})$ is defined by $twom(n) = \min(0, 2 - n)$

Using the usual ordering of the natural numbers \mathbb{N} , show that Tarski's fixpoint construction does not find the fixpoint (1) of this function.

What assumption needed for Tarski's fixpoint construction is violated?

The lifted natural numbers \mathbb{N}_\perp are an ordered set containing all natural numbers plus a special symbol \perp , where any natural number is greater than \perp and natural numbers are not related to each other in this ordering.

If you more comfortable with partial functions than with liftings, use partial functions instead.

For each of the following, (i) What is the least fixpoint of the function? (ii) Give a brief description of all fixpoints of the function.

c) (3 pts) Function i of type $(\mathbb{N}_\perp \rightarrow \mathbb{N}_\perp) \rightarrow (\mathbb{N}_\perp \rightarrow \mathbb{N}_\perp)$ defined by $i(f) = f$.

d) (3 pts) Function $strange$ of type $(\mathbb{N}_\perp \rightarrow \mathbb{N}_\perp) \rightarrow (\mathbb{N}_\perp \rightarrow \mathbb{N}_\perp)$ defined by $(strange(f))(n) = \text{if } n \geq 4 \text{ then } 0 \text{ else } f(n \times 2)$

5. Environments of evaluation. (10 pts)

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 calc3
  (lambda (proc a b c)
    (proc a b c)))
```

```
(define product
  (lambda (args)
    (apply * args)))
```

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

```
(calc3 product 2 3 4)
```

6. Problem: Prolog programming. (10 pts)

Given the Prolog definition of merge (variables begin with Upper Case letters):

```
merge([X|Xs], [Y|Ys], [X|Zs]) :- X < Y, merge(Xs, [Y|Ys], Zs).
merge([X|Xs], [Y|Ys], [X, Y |Zs]) :- X = Y, merge(Xs, Ys, Zs).
merge([X|Xs], [Y|Ys], [Y|Zs]) :- X > Y, merge([X|Xs], Ys, Zs).
merge(Xs, [], Xs).
merge([], Ys, Ys).
```

a) (5 pts) What is the result of the query `merge([1, 3, 6], [2, 4, 5], Z)`? Show the results of the successive applications and the variable bindings produced by unification as the program executes.

b) (5 pts) Note that, if the goal $X < Y$ succeeds, there is no reason to try any of the remaining clauses. In fact, success of any one of the clauses implies that none of the subsequent clauses should be tried. How can you express this in terms of the Prolog "cut"? What are the semantics of the "cut"? (i.e., what does the "cut" do?)

7. Curried functions and types. (10 pts)

a) (2 pts: ML) Write a function `map` that takes a function $F : a \rightarrow b$ as argument and produces a function that takes a list of elements of type a and returns the corresponding list of their images under F .

b) (2 pts: ML) Write an uncurried version of `map`.

c) (2 pts: Scheme) Write a function `map` that takes a function $F : a \rightarrow b$ as argument and produces a function that takes a list of elements of type a and returns the corresponding list of their images under F .

d) (2 pts: Scheme) Write an uncurried version of `map`.

e) (2 pts) What can you say about the type constraints that `map` has to satisfy in the two languages?

8. Types. (10 pts)

a) (3 pts) Define ad-hoc polymorphism

b) (7 pts) Two major cases of ad-hoc polymorphism are *overloading* and *coercion*. Explain, and provide examples in two of 1) a functional language; 2) an imperative language; 3) an object-oriented language.

9. Programming in the large. (10 pts)

Describe the notion of *encapsulation*, and describe encapsulation mechanisms provided by C, C++ and ML. Include a brief discussion of safety (i.e., the ability or inability of a client to defeat the encapsulation mechanism provided).

10. Problem: ML Polymorphism. (10 pts)

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);`