

CS Qualifying Exam

Fall 2002

Programming Languages

by Adriana Compagnoni

September 2002

Important remarks

- This test has 5 questions of 20 points each.
- The grading scale is: A= 90-100, B= 80-90, C=70-80, D=60-70, else Fail.
- This test is closed book, closed notes, and closed laptops.
- If you find any question unclear, please specify your understanding of what is being asked.

Subset of Scheme You May Use

Unless otherwise specified, when defining procedures you may only use: helping procedures that you define yourself, comments, and the procedures and keywords that are included in the following list. The notation `c...r` means `caar`, `cadr`, `cddr`, etc.

```
', #t, #f, *, +, -, /, <, <=, =, >=, >,
and, andmap, append, apply, boolean?,
car, cdr, c...r, char? cond, cons,
define, display, else, eq? equal?, eqv?, error, if, let, letrec,
lambda, list, length, list?, map, newline, not, null?, number?,
or, pair?, procedure?, quote,
string, string?, string=?, string-append,
string-ci=?, string-length, string-ref,
string->list, string->number,
string->symbol, substring, symbol?
vector, vector?, vector-length,
vector->list, vector-ref, zero?
define-datatype, cases.
```

1. (a) Write the BNF <BT> for binary trees with numbers in the leaves and in the internal nodes.
 (b) Write a syntactic derivation that proves that (6 1 (2 3 5)) is a <BT>.
 (c) Define the corresponding Scheme datatype bin-tree.
 (d) Write a procedure (sum-tree bt), which given a bin-tree bt returns another bin-tree containing in each node the sum of its subtree.

For example,

```
> (sum-tree '(6 1 (2 3 5)) )
(17 1 (10 3 5))
> (sum-tree '(1 3 4) )
(8 3 4)
```

2. Consider the following grammar defining the concrete syntax of <exp>

```
<exp> ::= <integer>
        | <boolean>
        | <identifier>
        | (if <exp> <exp> <exp>)
        | (and <exp>+)
        | (or <exp>+)
        | (not <exp>)
```

Given (if eb et ef) we say that et is a *true branch* and ef is a *false branch*. Prove by induction that any expression <exp> has the same number of true branches as false branches.

3. Given the following grammar defining the concrete syntax of <exp> and the corresponding datatype defining the abstract syntax:

```
<exp> ::= <integer>
        | <boolean>
        | <identifier>
        | (if <exp> <exp> <exp>)
        | (and <exp>+)
        | (or <exp>+)
        | (not <exp>)
```

```
(define-datatype exp exp?
  (lit-exp (datum number?))
  (bool-exp (bool boolean?))
  (var-exp (var symbol?))
  (if-exp (test exp?)
          (then exp?)
          (else exp?))
  (and-exp (args (list-of exp?)))
  (or-exp (args (list-of exp?)))
  (not-exp (arg)))
```

Write an interpreter eval e env that evaluates expression e in environment env

4. Write half a page describing the phases of a compiler.
5. Write an imperative program that has different side effects when evaluated under the call-by-value and the call-by-reference parameter-passing mechanisms.