Principles of Programming Languages - Final

K. Subramani
LDCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Instructions

(a) The final is to be turned in by 10:30 am.
(b) Each question is worth 4 points.
(c) Attempt as many questions as you can; you will be given partial credit.

2 Problems

1. Syntax:
Consider the context-free grammar \( G = (V, T, P, S) \), where \( V = \{S\} \), \( T = \{0, 1\} \), and the productions \( P \) are defined by:

\[
S \rightarrow 0 S 1 \mid 1 S 0 \mid \epsilon
\]

Argue that every string generated by this grammar is balanced, i.e., if \( w \) is derived from \( S \), then \( n_0(w) = n_1(w) \), where \( n_0(w) \) and \( n_1(w) \) stand for the number of 0s and 1s respectively in \( w \).

2. Semantics:
   (a) Explain with an example the difference between the scope of a binding and its visibility. What is a scope-hole? (1 point.)
   (b) Explain the difference between storage semantics and pointer semantics when it comes to variable assignment. (1 point.)
   (c) Which of the following expressions is an l-value in C:
       (i) \&(*x + 1).
       (ii) *(&x + 1).

3. Procedures and Environments:
   (a) Provide a brief description of the working of the mark-and-sweep algorithm with respect to storage reclamation, identifying its drawbacks. (3 points.)
   (b) Describe one improvement to the mark-and-sweep framework that has been implemented in current memory management systems.
4. **Functional Programming:**

(a) In class, we discussed a SCHEME function to reverse a list; that function reverses only the top level of the input list. For instance, consider the list ((1 2) 3 (4 5)); the function discussed in class would produce the list ((4 5) 3 (1 2)) on reversal. Write a SCHEME function called `deep-reverse` that recursively reverses all the sublists of an input list; for instance, the deep-reversal of the above list should return ((5 4) 3 (2 1)). You may assume the existence of the `append()` function in SCHEME which takes two lists L and M as input and returns a list which contains all the elements of L followed by all the elements of M, in precisely the same order as they occur in L and M. (3 points.)

(b) Consider the following expression in SCHEME:

```scheme
(let ((x 2) (y 3) E)
  ...
)
```

where E is an arbitrary expression. Rewrite the above expression as a lambda application without using `let`.

5. **Logic Programming:**

(a) Let P, Q and R be propositions. Prove the validity of the following argument without using truth-tables:

\[ P \land (Q \lor R) \rightarrow [(P \land Q) \lor (P \land R)] \]

You may assume the following tautology called the Deduction Method:

\[ [(A \land B) \rightarrow (C \rightarrow D)] \leftrightarrow (A \land B \land C) \rightarrow D \] (2 points.)

(b) Write a Prolog fragment that reverses a list, using pattern-directed invocation. You may assume that the list is composed of atoms only.