UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS 263 **Spring 1996**
K. A. Yelick

CS 263: Design of Programming Languages
Homework 5
Due April 16, 1996

1. Axiomatic Semantics.

Consider the following program written in FL! with implicit references and infix assignment statements. For your convenience, the program is "commented" by a version of the same program in the right column written in pseudo-C.

Using either pre-post style axiomatic semantics or wp-semantics, prove that the program will terminate in a state in which `ans` is true iff and only if `sum` is divisible by 3, i.e., `(= ans (mod sum 3) 0)`. Assume that `mod` is a built-in primitive and assume any arithmetic axioms that are needed.

```
(begin
  (s := sum)                          s = sum;
  (while (>= s 10)                    while s >= 10 {
      (x := (/ s 10))                     x = s div 10;
      (y := (- s (* 10 x)))               y = s - 10*x;
      (s := (+ x y))                      s = x + y;
  )                                       }
  (if (= (mod s 3) 0)                 if s mod 3 == 0
      (ans := true)                       ans = true;
                                        else
      (ans := false)                      ans = false;
  )
)
```

2. Loop Invariants

Given an array `x[1..n]` of extended integers (i.e., integers plus infinities). What is the interesting property of `p` that holds when the following program terminates? (Hint: run some example and observe `y`.)

```
array x[1..n], y[1..n] of extended_integer;

for j = 1 to n
    y[j] = +infinity;
p = 0;
```

```
for i = 1 to n {                    % outer loop
    for j = 1 to p+1                % inner loop
        if  x[i] < y[j] {
            y[j] = x[i];
            break;
        }
    if j > p
        p = j;
}
```

What are the loop invariants for the two loops labeled `outer loop` and `inner loop` that you would using in proving the property holds.

3. Standard Semantics

Louis Reasoner has been hired by Structured Software Inc. to develop a standard denotational semantics for the SFL! (Structured FL!) an extension of FL! with the following additional expressions:

`E ::= (for I` $\mathtt{E}_{low}$ $\mathtt{E}_{high}$ $\mathtt{E}_{body}$`) | (exit` $\mathtt{E}_{levels}$`)`

Informally, the semantics of these constructs are:

- `(for I` $\mathtt{E}_{low}$ $\mathtt{E}_{high}$ $\mathtt{E}_{body}$`)` is a for loop, similar to one that you would find in other languages. Both $\mathtt{E}_{low}$ and $\mathtt{E}_{high}$ are evaluated (in order). Both must evaluate to integer values. The body $\mathtt{E}_{body}$ is then evaluated $\mathtt{E}_{high}$-$\mathtt{E}_{low}$+1 times, with `I` bound to $\mathtt{E}_{low}$, $\mathtt{E}_{low}$`+1`, ... , $\mathtt{E}_{high}$. If $\mathtt{E}_{high}$ is less than $\mathtt{E}_{low}$, the loop body will not execute at all. Note that `I` is an integer, not a `ref`, so it cannot be modified in the loop body, and `I` can only be used in an expression without a `^` dereference. The value of a `for` loop is the unit value.
- `(exit` $\mathtt{E}_{levels}$`)` is a multilevel exit. The value of $\mathtt{E}_{levels}$ must be a positive integer. The effect is to jump out of $\mathtt{E}_{levels}$ of nested `for` loops, and continue executing after the loop. An error results if there are less than $\mathtt{E}_{levels}$ levels of nested `for` loops at the `exit` point. (The value of the loop is still the unit value when exited by an `exit`.)

Louis has decided to add a new domain to the semantics of FL! to represent a stack of expression continuations for the exit points defined by nested `for` loops.

$q \in ContStack = ExpCont^*$

a. Make any modification the semantic domains to handle the *ContStack* domain.

b. Give the valuation function for `for` loops. (Hint: `for` loops as defined are much less general than `while` loops or recursion. Although the loop bounds are not "compile-time" constants, think about how you might desugar `for` loops if they were constants. Write one or two helping functions.)

c. Give the valuation function for `exit`.

Note: We believe that the solution using a sequence domain for ContStack is fairly clean, but other domains will work too. If you would rather use something else, you may do so without losing points. Just be sure to warn us.