Written Assignment 6

This assignment asks you to prepare written answers to questions on code layout and operational semantics. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

Please print your name and email address on your homework!

We need this information so that we can give you credit for the assignment and so that we can return it to you.

1. Consider the following piece of code:

```plaintext
fact(n : Int) : Int {
    if n > 0 then n*fact(n-1) else 1 fi
} ;
```

Draw the tree of activation records for a call to `fact(4)`.

2. Consider these six operational semantics rules:

   (1) \[
   \text{so, } E, S \vdash e_1 : \text{Bool}(false), S_1 \\
   \text{so, } E, S \vdash \text{while } e_1 \text{ loop } e_2 \text{ pool : void}, S_1
   \]

   (2) \[
   \text{so, } E, S \vdash e_1 : \text{Bool}(true), S_1 \\
   \text{so, } E, S_1 \vdash e_2 : v, S_2
   \]

   (3) \[
   \text{so, } E, S \vdash \text{let } id : T \leftarrow e_1 \text{ in } e_2 : v_2, S_2 \\
   l_{\text{new}} = \text{newloc}(S_1)
   \]

   (4) \[
   E(id) = l_{id} \\
   S(l_{id}) = v \\
   \text{so, } E, S \vdash id : v, S
   \]

   (5) \[
   E(id) = l_{id} \\
   S_2 = S_1[v/l_{id}] \\
   \text{so, } E, S \vdash id \leftarrow e : v, S_2
   \]

   (6) \[
   v = \begin{cases} 
   \text{Bool(true)} & \text{if } n_1 < n_2 \\
   \text{Bool(false)} & \text{if } n_1 \geq n_2 
   \end{cases}
   \]

   \[
   \text{so, } E, S \vdash e_1 < e_2 : v, S_2
   \]

Use these rules to construct a derivation for the following piece of code:
let x : Int <- 2 in
while 1 < x loop
  x <- x - 1
pool

You may assume reasonable axioms, e.g. it is always true that so, E, S ⊢ 2 − 1 : Int(1), S. Start your derivation using the let rule (3) as follows:

<table>
<thead>
<tr>
<th>so, E, S ⊢ 2 : Int(2), S</th>
<th>so, E[lnew/x], S[Int(2)/lnew] ⊢ while 1 &lt; x loop x ← x − 1 pool : void, S_final</th>
</tr>
</thead>
<tbody>
<tr>
<td>so, E, S ⊢ let x : Int ← 2 in while 1 &lt; x loop x ← x − 1 pool : void, S_final</td>
<td></td>
</tr>
</tbody>
</table>

Note that you only need to expand hypotheses that need to be proved (i.e. those containing ⊢).

3. Suppose we wanted to add arrays to COOL, using the following syntax:

let a:T[e] in e2
a[e] <- e2
a[e]

Create an array a with size e1 of Ts, usable in e2
Assign e2 to element e1 in a
Get element e of a

Write the operational semantics for these three syntactic constructs. You may find it helpful to think of an array of type T[n] as an object with n attributes of type T.

4. The operational semantics for COOL’s while expression show that result of evaluating such an expression is always void.

However, we could have used the following alternative semantics:

- If the loop body executes at least once, the result of the while expression is the result from the last iteration of the loop body.
- If the loop body never executes (i.e., the condition is false the first time it is evaluated), then the result of the while expression is void.

For example, consider the following expression:

while (x < 10) loop x <- x+1 pool

The result of this expression would be 10 if, initially, x < 10 or void if x ≥ 10. Write new operational rules for the while construct that formalize these alternative semantics.