Review Set 5

This Review Set asks you to prepare written answers to questions on runtime organization and operational semantics. Each of the questions has a short answer. You may discuss this Review Set with other students and work on the problems together.

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)`. Include at least the stack pointers, frame pointers, return values, and parameters.

2. Consider these six operational semantics rules:

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

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

(3) \[ \frac{\text{so, } E[l_{\text{new}}/id], S_1[v_1/l_{\text{new}}] \vdash e_2 : v_2, S_2}{\text{so, } E, S \vdash \text{let } id : T \leftarrow e_1 \text{ in } e_2 : v_2, S_2} \]

(4) \[ E(id) = l_{id} \]

(5) \[ S(l_{id}) = v \]

(6) \[ S_2 = S_1[v/l_{id}] \]

Use these rules to construct a derivation for the following piece of code:

```plaintext
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:

\[
\begin{align*}
\text{so}, E, S & \vdash 2 : \text{Int}(2), S \\
\frac{\text{so}, E[l_{\text{new}}/x], S[\text{Int}(2)/l_{\text{new}}] \vdash \text{while} \ 1 < x \ \text{loop} \ x \leftarrow x - 1 \ \text{pool} : \text{void}, S_{\text{final}}}{\text{so}, E, S \vdash x : \text{Int} \ - \ x - 1 \ \text{pool} : \text{void}, S_{\text{final}}} (2)
\end{align*}
\]

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:

\[
\begin{align*}
\text{let} \ a : T[e_1] \ \text{in} \ e_2 & \quad \text{Create an array } a \text{ with size } e_1 \text{ of } T, \text{usable in } e_2 \\
a[e_1] & \leftarrow e_2 \quad \text{Assign } e_2 \text{ to element } e_1 \text{ in } a \\
a[e] & \quad \text{Get element } e \text{ of } a
\end{align*}
\]

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:

\[
\text{while} \ (x < 10) \ \text{loop} \ x \leftarrow x + 1 \ \text{pool}
\]

The result of this expression would be 10 if, initially, \( x < 10 \) or void if \( x \geq 10 \).

Write new operational rules for the while construct that formalize these alternative semantics.