15-312 Foundations of Programming Languages

Midterm Examination

October 17, 2002

Name: ______________________________

Andrew User ID: ____________________

- This is a closed-book exam; only one double-sided sheet of notes is permitted.
- Write your answer legibly in the space provided.
- There are 11 pages in this exam, including 3 worksheets.
- It consists of 3 questions worth a total of 100 points and one extra credit question worth 20 points.
- The extra credit is recorded separately, so only attempt this after you have completed all other questions.
- You have 85 minutes for this exam.

Problem 1	Problem 2	Problem 3	Total	EC
45	**25**	**30**	**100**	**20**

1. Static and Dynamic Semantics (45 pts)

In order to add lazy evaluation uniformly and compositionally to a call-by-value language such as MinML we add a new type τ susp and constructs delay(e) and force(e). Informally, delay(e) delays the evaluation of e by creating a suspension, while force(e) forces the evaluation of a suspension. For the purpose of this problem, we do not memoize suspensions.

The basis for the question below are the judgments $\Gamma \vdash e : \tau$, e value and $e \mapsto e'$, the latter from a structured operational semantics.

1. (5 pts) Show the typing rules for delay and force using the new type τ susp.

2. (10 pts) Show the new rules for values and the transition rules for the new constructs in a structured operational semantics.

3. (5 pts) State the new case for the value inversion lemma (also known as the canonical forms lemma). You do not need to prove it.

4. (5 pts) State the type preservation theorem in its form appropriate for the judgments considered here. It is proved by rule induction over which derivation?

5. (10 pts) Show the cases for the new constructs in the proof of the type preservation theorem.

6. (10 pts) Assume that, in addition to suspensions, our language already has (non-recursive) call-by-value functions $\tau_1 \rightarrow \tau_2$ with abstraction $\lambda x.e$ and application $e_1\, e_2$. We would like to add a call-by-name function space $\tau_1 \Rightarrow \tau_2$ with abstraction $\lambda_{\mathsf{n}} x.e$ and application $e_1 \cdot e_2$. Show how the call-by-name constructs can be considered syntactic sugar by giving their expansion. You do not need to prove correctness.

$\tau_1 \Rightarrow \tau_2$ expands to ______________________________

$\lambda_{\mathsf{n}} x.e$ expands to ______________________________

$e_1 \cdot e_2$ expands to ______________________________

2. Continuations (25 pts)

Assume we are in the MinML language extended with continuations, but not exceptions.

1. (10 pts) Explain, in words, the behavior of the continuation k returned by the following function when given two types and a pair of continuations (k_1, k_2).

```
Fun a in Fun b in
  fun join (p:a cont * b cont): (a + b) cont is
    letcc ret in
     case (letcc r in throw r to ret end)
       of inl(x1) => throw x1 to fst p
        | inr(x2) => throw x2 to snd p
      end
    end
  end
end end
```

2. (5 pts) Call the function above `polyjoin` and let

$$\texttt{ijoin} = \texttt{polyjoin [int] [int]}.$$

What is the type of `ijoin`?

3. (5 pts) What is the value of the following expression?

```
1 + letcc k1 in 2 + letcc k2 in
  throw inl(3) to ijoin (k1, k2)
end end
```

4. (5 pts) What is the value of the following expression?

```
1 + letcc k1 in 2 + letcc k2 in
  throw inr(3) to ijoin (k1, k2)
end end
```

5. (20 pts EXTRA CREDIT) Write a function

```
polysplit : All a. All b. (a + b) cont -> a cont * b cont
```

that achieves the converse of the `polyjoin` function above. Explain in words what this function accomplishes.

3. Recursive Types (30 pts)

Consider the ML data type

```
datatype Tree = Leaf | Unary of Tree | Binary of Tree * Tree
```

1. (5 pts) Give a definition of `Tree` as a recursive type using the $\mu t.\tau$ construct.

2. (5 pts) Give the definition of the constructors `Leaf`, `Unary`, and `Binary`.

3. (10 pts) On your representation, write a function `count` : `Tree` → `int` that counts the number of leaves in a given tree.

4. (10 pts) Give a definition of `Tree` as an abstract type. Your operations on the type should just be the three constructors and a destructor in the form of a case expression for trees. You do not need to give an implementation of the abstract type.

Worksheet

Worksheet

Worksheet