Midterm Exam
CS6848

Maximum marks = 40, Time: 2hrs

23-Sep-2015

Read all the instructions and questions carefully. You can make any reasonable assumptions that you think are necessary; but state them clearly. There are total four questions totaling 40 marks (+ 2 Bonus). Each five marks will approximately take 15 minutes. For questions that have sub-parts, the division for the sub-parts are mentioned in square brackets.

Leave the first page empty. Start each question on a new page. Think about the question before you start writing and write briefly. **The answer for any question (including all the sub-parts) should NOT cross more than two pages.** If the answer is spanning more than two pages, we will ignore the spill-over text. If you scratch/cross some part of the answer, you can get compensation space from the next page.

1. [10+2] **Scheme, Programming, Correctness**

   (a) What is the output of the following scheme code and Show the evaluation. [Bonus, 2]

```
(define Y1
  (lambda (f)
    ((lambda (x)
      (f (x x)))
     (lambda (x)
       (f (x x))))))
((Y1 fact) 5)
```

   (b) What is the output of the following scheme code? Show the evaluation [4].

```
(define fact
  (lambda (f)
    (lambda (n)
      (if (zero? n) 1 (* n (f (- n 1)))))))

(define Y2
  (lambda (h)
    ((lambda (x) (x x))
     (lambda (g)
       (h (lambda (args)
            ((g g) args)))))))
((Y2 fact) 5)
```

(c) [6] For the fibonacci function:

$$F = \lambda f.\lambda n.\ if\ (<\ n\ 2)\ n\ (+\ (f\ (-\ n\ 1))\ (f\ (-\ n\ 2)))$$

show that $((YF)\ n)$ computes fibbonaci(n), where Y is the Y-combinator.

2. [15] **Type soundness**: Extend the simply typed lambda calculus (no integers) with booleans and if-else expression. Show the extended grammar, values, and types [5]. Write the type rules, and operational semantics (small-step) [5]. Prove the progress lemma [5]. Feel free to state and use the other standard lemmas (Useless Assumption, Substitution, Type Preservation, and Typable Value), as axioms.

3. [10] **Recursive types and Subtyping**

   (a) Argue that the unification algorithm studied in the class terminates for both simple and recursive types. [1]. Argue that the generated unification is the most general solution [2].

   (b) State the Arrow rule for subtyping (for subtyping of function types) [1] and argue why the choice of covariant and contravariant subtyping makes sense [2].

   (c) Answer if the following is true (by constructing the product automata) [4]:

$$(\mu u.(u \to (\mu v.(v \to \top))) \to u) \leq (\mu s.(\bot \to (\mu w.(w \to s))) \to \bot)$$

4. [5] **Semantics** Write the big step and small step rules for simply typed lambda calculus, extended with integers and `succ` operator, assuming lazy evaluation.