Koç University
College of Engineering

COMP 301: Programming Language Concepts

Midterm Examination
Nov 7, 2017, Tuesday 19:00-20:30
SOS B10 - SOS B11

Instructor: T. Metin Sezgin
Time Allowed: 120 minutes

Name:___________________________

Student Number:________________

NOTE: Explain your answers in full. Provide all the work in your exam paper, but make sure the answer boxes have nothing but your final answer to the questions. Include signatures (contracts) for all scheme functions that you implement.

I pledge on my honor that I have neither given nor received unauthorized assistance on this exam.

Signature:________________

Question	Points	Grade
1	10	
2	10	
3	30	
4	10	
5	20	
6	10	
7	10	
BONUS	10	
Total	110	

1. (10 points)

   In the class we covered top-down and bottom-up strategies for representing sets. At the core, both techniques describe sets by relating their members to the base case. They differ fundamentally on how they accomplish this. In 20 words or less, describe the nature of this difference.

   Answer (20 words max):

   Word count:

2. (10 points)

   In the class, we designed laser-cut jigsaw pieces for representing LcExp and LET expressions. Proc extends LET by two new expressions. In the space below, draw and label the shapes for these two new expressions.

   Answer:

   Word count:

3. (30 points) Consider a new expression `let-if` that combines the semantics of `let` and `if` such that the body of the `let` is evaluated by setting a variable based on the outcome of a test. For example,

```
let-if x = 5 or 10 based-on zero?(0)
in -(x,2) or -(x, 4)
```

returns 3, and

```
let-if x = 5 or 10 based-on z
in -(x,2) or -(x, 4)
```

returns 6 if z is false.

(a) (2 points) Write down the concrete grammar entry for this new expression.

Answer:

(b) (3 points) Write down the abstract syntax representation. [1]

Answer:

[1]**Hint:** This is what we put inside rectangular boxes when we talked about abstract syntax.

(c) (5 points) Give the `define-datatype` entry for `let-if`.

Answer:

(d) (10 points) Provide the behavior specification for the new expression.

Answer:

(e) (10 points) Give the implementation (just the bit that would go inside `value-of`).

Answer:

4. (10 points) The unparse-lc-exp procedure given below produces a lambda calculus expression from an abstract syntax tree representation.

```
(define unparse-lc-exp
  (lambda (exp)
    (cases lc-exp exp
      (var-exp (var) var)
      (lambda-exp (bound-var body)
        (list 'lambda (list bound-var)
          (unparse-lc-exp body)))
      (app-exp (rator rand)
             ------------------------
     )
    )))
```

   (a) (5 points) Fill in the blank.

   (b) (5 points) Write down the signature (contract) for this procedure.

   Answer:

5. (20 points) Consider the following program.

```
let x = proc (y) -(y,1)
     in let y = proc(x) -(x,2)
          in (y (x (y 10)))
```

(a) (10 points) Draw the corresponding abstract syntax tree.

(b) (10 points) What does the expression evaluate to?

6. (10 points) Consider the following expression:

```
let a = 1
  in let f = proc(x) -(x,1)
    in let x = 2
      in (f 10)
```

Show the state of the environment during the evaluation of this program at the specific time when `value-of` gets the expression `<<-(x,1)>>`. You can show the environment using the textual representation or draw it using the box and pointer representation as shown in the class and in the textbook.

Answer:

7. (10 points) Consider a variant of PROC that only allows the creation of anonymous procedures, but doesn't allow us to name them. In this new variant, we are also not allowed to pass procedures as inputs to other procedures.

   List the set of denoted and expressed values for this new language.

Answer:

8. (10 points) BONUS Provide a functional representation that implements pairs through `car`, `cdr`, and `cons`. [2]

[2]No partial credit on this question. Do not waste time on this question unless you are sure that you will get it all correct.