COP-4020

PROGRAMMING LANGUAGE CONCEPTS

Instructor: Manuel Bermudez

SPRING 2011

FINAL EXAM

Take-home, due April 28 5 p.m.

TURN IN YOUR SOLUTION AS A SINGLE, PLAIN TEXT FILE WITH YOUR ENHANCED TINY INTERPRETER, WRITTEN IN RPAL.
PROBLEM 1. (10 points)

Add the following binary operators to the RPAL implementation of Tiny, as discussed in class. For your reference, the RPAL syntax grammar is shown in Appendix A.

' >= ', '< ', '> ', '=' , '< > ', '-' , '*' , '/' , ' and ', ' or '.

PROBLEM 2. (10 points)

Add the following unary operators to the RPAL implementation of Tiny, as discussed in class.

' - ', '+ '.

PROBLEM 3. (10 points)

Add the 'cond' construct (conditional expression) to Tiny, as described in class.

PROBLEM 4. (10 points)

Add the prefix and postfix auto-increment operators to Tiny, as described in class.

PROBLEM 5. (10 points)

Add the one-armed 'if' statement to Tiny.

PROBLEM 6. (10 points)

Add the 'repeat' statement to Tiny.

PROBLEM 7. (10 points)

Add the 'read' statement to Tiny.
PROBLEM 8. (10 points)
Add the 'for' statement (as in the C language) to Tiny.

PROBLEM 9. (20 points)
Add the 'case' statement to Tiny. Implement the semantics of Pascal: after a match between the case statement’s control expression and a case label, the corresponding statement is executed, and the case statement aborts.
APPENDIX: RPAL’s Phrase Structure Grammar:

# Expressions ##################################################################

E  -> 'let' D 'in' E  => 'let'
    -> 'fn' Vb+ '.' E  => 'lambda'
    -> Ew;
Ew -> T 'where' Dr   => 'where'
    -> T;

# Tuple Expressions ##################################################################

T  -> Ta (',' Ta )+ => 'tau'
    -> Ta ;
Ta -> Ta 'aug' Tc   => 'aug'
    -> Tc ;
Tc -> B '->' Tc '|' Tc => '->'
    -> B ;

# Boolean Expressions ##################################################################

B  -> B 'or' Bt => 'or'
    -> Bt ;
Bt -> Bt '&' Bs => '&'
    -> Bs ;
Bs -> 'not' Bp => 'not'
    -> Bp ;
Bp -> A ('gr' | '>=' ) A => 'gr'
    -> A ('ge' | '>=' ) A => 'ge'
    -> A ('ls' | '<=' ) A => 'ls'
    -> A ('le' | '<=' ) A => 'le'
    -> A 'eq' A => 'eq'
    -> A 'ne' A => 'ne'
    -> A ;

# Arithmetic Expressions ##################################################################

A  -> A '+' At => '+'
    -> A '-' At => '-'
    -> '+' At
    -> '-' At => 'neg'
    -> At ;
At -> At '**' Af => '**'
    -> At '/' Af => '/'
    -> Af ;
Af -> Ap '***' Af => '***'
    -> Ap ;
Ap -> Ap '@' '<IDENTIFIER>' R => '@'
    -> R ;
# Rators And Rands ###################

```
R  -> R Rn => 'gamma'
  -> Rn ;
Rn -> '<IDENTIFIER>'
  -> '<INTEGER>'
  -> '<STRING>'
  -> 'true' => 'true'
  -> 'false' => 'false'
  -> 'nil' => 'nil'
  -> 'dummy' => 'dummy' ;
```

# Definitions ################################

```
D  -> Da 'within' D => 'within'
  -> Da ;
Da -> Dr ( 'and' Dr )+ => 'and'
  -> Dr ;
Dr -> 'rec' Db => 'rec'
  -> Db ;
Db -> Vl '=' E => '='
  -> '<IDENTIFIER>' Vb+ '=' E => 'fcn_form'
  -> '(' D ')' ;
```

# Variables ################################

```
Vb -> '<IDENTIFIER>'
  -> '(' Vl ')' => '(')
  -> ')' => ')' ;
Vl -> '<IDENTIFIER>' list ',' => ',' ;
```