19CSE401 - Compiler Design

Programming Language : Racket

Name	**Roll Number**
Raghul K B	CB.EN.U4CSE19346
O Mahanth	CB.EN.U4CSE19340
V Devakumar	CB.EN.U4CSE19358
V Nithin Krishna	CB.EN.U4CSE19360

CFG:

```
start : program
program : defOrExpr program EOF
program : defOrExpr ;

defOrExpr : definition
defOrExpr : expr

nameplus : name nameplus
nameplus: name ;

namestar : name namestar
namestar : ε ;

definitionstar : definition definitionstar
definitionstar : ε ;

definition : LEFTB DEFINE name expr RIGHTB
definition : LEFTB DEFINE LEFTB name nameplus RIGHTB expr RIGHTB
definition : LEFTB DEFINESTRUCT name LEFTB namestar RIGHTB RIGHTB ;

exprplus : expr exprplus
exprplus : expr ;

lner : LEFTSQB name expr RIGHTSQB lner
lner : ε ;

leerbplus : LEFTB expr expr RIGHTB leerbplus
leerbplus : LEFTB expr expr RIGHTB ;
              ;
```

```
leersqbplus : LEFTSQB expr expr RIGHTSQB leersqbplus
leersqbplus : LEFTSQB expr expr RIGHTSQB ;

leersqbstar : LEFTSQB expr expr RIGHTSQB leersqbstar
leersqbstar : ε ;

expr : LEFTB BEGINN exprplus RIGHTB
expr : LEFTB BEGINN0 exprplus RIGHTB
expr : LEFTB SETNQ NAME expr RIGHTB
expr : LEFTB DELAY expr RIGHTB
expr : LEFTB CAR expr RIGHTB
expr : LEFTB CDR expr RIGHTB
expr : LEFTB COMBINATIONS expr RIGHTB
expr : LEFTB LIST expr RIGHTB
expr : LEFTB REVERSE expr RIGHTB
expr : LEFTB APPEND NAME expr RIGHTB
expr : LEFTB LAMBDA LEFTB namestar RIGHTB expr RIGHTB
expr : LEFTB LAMBDASYM LEFTB namestar RIGHTB expr RIGHTB
expr : LEFTB LOCAL LEFTSQB definitionstar RIGHTSQB expr RIGHTB
expr : LEFTB LETREC LEFTB lner RIGHTB expr RIGHTB
expr : LEFTB SHARED LEFTB lner RIGHTB expr RIGHTB
expr : LEFTB LET LEFTB lner RIGHTB expr RIGHTB
expr : LEFTB LETSTAR LEFTB lner RIGHTB expr RIGHTB
expr : LEFTB RECUR name LEFTB lner RIGHTB expr RIGHTB
expr : LEFTB name exprplus RIGHTB
expr : LEFTB COND leerbplus RIGHTB
expr :LEFTB COND leersqbplus RIGHTB
expr : LEFTB COND leersqbstar LEFTSQB ELSE expr RIGHTSQB RIGHTB
expr : LEFTB IF  expr expr expr RIGHTB
expr : LEFTB AND expr exprplus RIGHTB
expr : LEFTB OR  expr exprplus RIGHTB
expr : DISPLAY name
expr :DISPLAY string
expr : NEWLINE
expr : EMPTY
expr : QUOTESX
expr : QSMARK
expr :QUOTEQUOTED quoted
expr :QUOTEQUASIQUOTED quasiQuoted
expr :CHARACTERQUOTED
expr : reloperators
expr :arthoperators
expr :name
expr : number
expr : symbol
expr : bool
expr :string
expr : character;


pkg : LEFTB string string number number RIGHTB;

name : NAME;

symbol : SYMBOL;

number : INT|DECIMAL;

bool : BOOLEAN;

string : STRING;

character : CHARACTER ;

reloperators : RELOPERATORS;

arthoperators : ARTHOPERATORS;
```

CFG	Names
program: defOrExpr program EOF	defOrExpr_program_eof_program
program: defOrExpr	defOrExpr_program
defOrExpr : definition	definition_defOrExpr
defOrExpr : expr	expr_defOrExpr
nameplus : name nameplus	name_nameplus_nameplus
namestar : name namestar	name_namestar_namestar
namestar : ε	epsilon_namestar
definitionstar : definition definitionstar	definition_definitionstar_definitionstar
definitionstar : ε	epsilon_definitionstar
definition : *LEFTB DEFINE* name expr *RIGHTB*	lb_def_name_expr_rb_definition
definition : *LEFTB DEFINESTRUCT* name *LEFTB* namestar *RIGHTB RIGHTB*	lb_def_lb_name_nameplus_rb_expr_rb_definition
exprplus : expr exprplus	expr_exprplus_exprplus

lner : *LEFTSQB* name expr *RIGHTSQB* lner	lsqb_name_expr_rsqb_lner_lner
lner : ε	epsilon_lner
leerbplus : *LEFTB* expr expr *RIGHTB* leerbplus	lb_expr_expr_rb_leerbplus_leerbplus
leerbplus : *LEFTB* expr expr *RIGHTB*	lb_expr_expr_rb_leerbplus
leersqbplus : *LEFTSQB* expr expr *RIGHTSQB* leersqbplus	lsqb_expr_expr_rsqb_leersqbplus_leersqbplus
leersqbplus : *LEFTSQB* expr expr *RIGHTSQB*	lsqb_expr_expr_rsqb_leersqbplus
leersqbstar : *LEFTSQB* expr expr *RIGHTSQB* leersqbstar	lsqb_expr_expr_rsqb_leersqbstar_leersqbstar
leersqbstar : ε	epsilon_leersqbstar
expr: *LEFTB BEGINN* exprplus *RIGHTB*	lb_begin_exprplus_rb_expr
expr: *LEFTB BEGINN0* exprplus *RIGHTB*	lb_begin0_exprplus_rb_expr
expr: *LEFTB SETNQ NAME* expr *RIGHTB*	lb_setnq_name_expr_rb_expr
expr: *LEFTB DELAY* expr *RIGHTB*	lb_delay_expr_rb_expr
expr: *LEFTB CAR* expr *RIGHTB*	lb_car_expr_rb_expr
expr:*LEFTB CDR* expr *RIGHTB*	lb_cdr_expr_rb_expr
expr: *LEFTB COMBINATIONS* expr *RIGHTB*	lb_combinations_expr_rb_expr
expr: *LEFTB LIST* expr *RIGHTB*	lb_list_expr_rb_expr
expr: *LEFTB REVERSE* expr *RIGHTB*	lb_reverse_expr_rb_expr
expr:*LEFTB APPEND NAME* expr *RIGHTB*	lb_append_name_expr_rb_expr
expr: *LEFTB LAMBDA LEFTB* namestar *RIGHTB* expr *RIGHTB*	lb_lambda_lb_namestar_rb_expr_rb_expr

expr: *LEFTB LAMBDASYM LEFTB* namestar *RIGHTB* expr *RIGHTB*	lb_lambdasym_lb_namestar_rb_expr_rb_expr
expr: *LEFTB LOCAL LEFTSQB* definitionstar *RIGHTSQB* expr *RIGHTB*	lb_local_lsqb_definitionstar_rsqb_expr_rb_expr
expr:*LEFTB LETREC LEFTB* lner *RIGHTB* expr *RIGHTB*	lb_letrec_lb_definitionstar_rsqb_expr_rb_expr
expr: *LEFTB SHARED LEFTB* lner *RIGHTB* expr *RIGHTB*	lb_shared_lb_lner_rb_expr_rb_expr
expr: *LEFTB LET LEFTB* lner *RIGHTB* expr *RIGHTB*	lb_let_lb_lner_rb_expr_rb_expr
expr:*LEFTB LETSTAR LEFTB* lner *RIGHTB* expr *RIGHTB*	lb_letstar_lb_lner_rb_expr_rb_expr
expr: *LEFTB RECUR* name *LEFTB* lner *RIGHTB* expr *RIGHTB*	lb_recur_name_lb_lner_rb_expr_rb_expr
expr:*LEFTB* name exprplus *RIGHTB*	lb_name_exprplus_rb_expr
expr: *LEFTB COND* leerbplus *RIGHTB*	lb_cond_leerbplus_rb_expr
expr:*LEFTB COND* leersqbplus *RIGHTB*	lb_cond_leersqbplus_rb_expr
expr: *LEFTB COND* leersqbstar *LEFTSQB ELSE* expr *RIGHTSQB RIGHTB*	lb_cond_leersqbstar_lsqb_else_expr_rsqb_rb_expr
expr: *LEFTB IF* expr expr expr *RIGHTB*	lb_if_expr_expr_expr_rb_expr
expr: *LEFTB AND* expr exprplus *RIGHTB*	lb_and_expr_exprplus_rb_expr
expr: *LEFTB OR* expr exprplus *RIGHTB*	lb_or_expr_exprplus_rb_expr

expr: *DISPLAY* name	Display_name_expr
expr:*DISPLAY* string	display_string_expr
expr: *NEWLINE*	newline_expr
expr: *EMPTY*	empty_expr
expr: *QUOTESX*	quotesx_expr
expr: *QSMARK*	qsmark_expr
expr:*CHARACTERQUOTED*	newline_expr
pkg: *LEFTB* string string number number *RIGHTB*;	lb_string_string_number_number_rb
name: *NAME*;	name_nameplus
symbol : *SYMBOL*;	symbol_expr
number : *INT*	int_expr
number : *DECIMAL*	decimal_expr
bool : *BOOLEAN*;	boolean_expr
string : *STRING*;	string_expr
character : *CHARACTER*	character_expr
reloperators : *RELOPERATORS*;	reloperators_expr
arthoperators : *ARTHOPERATORS*;	arthoperators_expr

ASTLexspec

```
grammar ASTspec;
@parser::header { import ast.*; }

start : t1= program {$t1.node.print();};

program returns [ASTNode node]: t1=defOrExpr t2=program EOF {$node = new
defOrExpr_program_eof_program($t1.node,$t2.node);}
                               | t3=defOrExpr {$node = new
defOrExpr_program($t3.node);};

defOrExpr returns [ASTNode node] :t1=definition {$node = new
definition_defOrExpr($t1.node);}
                                 | t2=expr {$node = new
expr_defOrExpr($t2.node);};

nameplus  returns [ASTNode node] : t1=name t2=nameplus {$node = new
name_nameplus_nameplus($t1.node,$t2.node);}
                                  | t3=name {$node = $t3.node};

namestar  returns [ASTNode node] : t1=name t2=namestar {$node = new
name_namestar_namestar($t1.node,$t2.node);}
                                     | {$node = new epsilon_namestar();}
                                     ;

definitionstar  returns [ASTNode node] : t1=definition t2=definitionstar
{$node = new definition_definitionstar_definitionstar($t1.node,$t2.node);}
                                             | {$node = new
epsilon_definitionstar();}
                                             ;

definition  returns [ASTNode node] : (LEFTB DEFINE t1=name t2=expr RIGHTB
{$node = new lb_def_name_expr_rb_definition($t1.node,$t2.node);})
                                   | (LEFTB DEFINE LEFTB t3=name t4=nameplus
RIGHTB t5=expr RIGHTB {$node = new
lb_def_lb_name_nameplus_rb_expr_rb_definition($t3.node,$t4.node,$t5.node);})
                                   | (LEFTB DEFINESTRUCT t6=name LEFTB
t7=namestar RIGHTB RIGHTB {$node = new
lb_defs_name_namestar_rb_rb($t6.node,$t7.node);});

exprplus  returns [ASTNode node] : t1=expr t2=exprplus {$node = new
expr_exprplus_exprplus($t1.node,$t2.node);}
                                     | t3=expr {$node = new
expr_exprplus($t3.node);};
```

```
lner  returns [ASTNode node] : (LEFTSQB t1=name t2=expr RIGHTSQB t3=lner
{$node = new lsqb_name_expr_rsqb_lner_lner($t1.node,$t2.node,$t3.node);})
                              | {$node = new epsilon_lner();}
                              ;

leerbplus  returns [ASTNode node] : (LEFTB t1=expr t2=expr RIGHTB t3=leerbplus
{$node = new
lb_expr_expr_rb_leerbplus_leerbplus($t1.node,$t2.node,$t3.node);})
                                     | (LEFTB t4=expr t5=expr RIGHTB {$node =
new lb_expr_expr_rb_leerbplus($t4.node,$t5.node);}) ;

leersqbplus  returns [ASTNode node] : (LEFTSQB t1=expr t2=expr RIGHTSQB
t3=leersqbplus {$node = new
lsqb_expr_expr_rsqb_leersqbplus_leersqbplus($t1.node,$t2.node,$t3.node);} )
                                   | (LEFTSQB t4=expr t5=expr RIGHTSQB {$node
= new lsqb_expr_expr_rsqb_leersqbplus($t4.node,$t5.node);});

leersqbstar  returns [ASTNode node] : (LEFTSQB t1=expr t2=expr RIGHTSQB
t3=leersqbstar {$node = new
lsqb_expr_expr_rsqb_leersqbstar_leersqbstar($t1.node,$t2.node,$t3.node);} )
                                          | {$node = new
epsilon_leersqbstar();}
                                          ;

expr  returns [ASTNode node]: (LEFTB BEGINN t1=exprplus RIGHTB {$node = new
lb_begin_exprplus_rb_expr($t1.node);})

    | (LEFTB BEGINN0 t2=exprplus RIGHTB {$node = new
lb_begin0_exprplus_rb_expr($t2.node);})

    | (LEFTB SETNQ NAME t3=expr RIGHTB {$node = new
lb_setnq_name_expr_rb_expr($t3.node);})

    | (LEFTB DELAY t4=expr RIGHTB {$node = new
lb_delay_expr_rb_expr($t4.node);})

    | (LEFTB CAR t5=expr RIGHTB {$node = new lb_car_expr_rb_expr($t5.node);})

    | (LEFTB CDR t6=expr RIGHTB {$node = new lb_cdr_expr_rb_expr($t6.node);})

    | (LEFTB COMBINATIONS t7=expr RIGHTB {$node = new
lb_combinations_expr_rb_expr($t7.node);})

    | (LEFTB LIST t8=expr RIGHTB {$node = new
lb_list_expr_rb_expr($t8.node);})
```

```
    | (LEFTB REVERSE t9=expr RIGHTB {$node = new
lb_reverse_expr_rb_expr($t9.node);})

    | (LEFTB APPEND NAME t10=expr RIGHTB {$node = new
lb_append_name_expr_rb_expr($t10.node);})

    | (LEFTB LAMBDA LEFTB t11=namestar RIGHTB t12=expr RIGHTB {$node = new
lb_lambda_lb_namestar_rb_expr_rb_expr($t11.node,$t12.node);})

    | (LEFTB LAMBDASYM LEFTB t13=namestar RIGHTB t14=expr RIGHTB {$node = new
lb_lambdasym_lb_namestar_rb_expr_rb_expr($t13.node,$t14.node);})

    | (LEFTB LOCAL LEFTSQB t15=definitionstar RIGHTSQB t16=expr RIGHTB {$node
= new lb_local_lsqb_definitionstar_rsqb_expr_rb_expr($t15.node,$t16.node);})

    | (LEFTB LETREC LEFTB t17=lner RIGHTB t18=expr RIGHTB {$node = new
lb_letrec_lb_definitionstar_rsqb_expr_rb_expr($t17.node,$t18.node);})

    | (LEFTB SHARED LEFTB t19=lner RIGHTB t20=expr RIGHTB {$node = new
lb_shared_lb_lner_rb_expr_rb_expr($t19.node,$t20.node);})

    | (LEFTB LET LEFTB t21=lner RIGHTB t22=expr RIGHTB {$node = new
lb_let_lb_lner_rb_expr_rb_expr($t21.node,$t22.node);})

    | (LEFTB LETSTAR LEFTB t23=lner RIGHTB t24=expr RIGHTB {$node = new
lb_letstar_lb_lner_rb_expr_rb_expr($t23.node,$t24.node);})

    | (LEFTB RECUR t25=name LEFTB t26=lner RIGHTB t27=expr RIGHTB {$node = new
lb_recur_name_lb_lner_rb_expr_rb_expr($t25.node,$t26.node,$t27.node);})

    | (LEFTB t28=name t29=exprplus RIGHTB {$node = new
lb_name_exprplus_rb_expr($t28.node,$t29.node);})

    | (LEFTB COND t30=leerbplus RIGHTB {$node = new
lb_cond_leerbplus_rb_expr($t30.node);})

    | (LEFTB COND t31=leersqbplus RIGHTB {$node = new
lb_cond_leersqbplus_rb_expr($t31.node);})

    | (LEFTB COND t32=leersqbstar LEFTSQB ELSE t33=expr RIGHTSQB RIGHTB {$node
= new lb_cond_leersqbstar_lsqb_else_expr_rsqb_rb_expr($t32.node,$t33.node);})

    | (LEFTB IF  t34=expr t35=expr t36=expr RIGHTB {$node = new
lb_if_expr_expr_expr_rb_expr($t34.node,$t35.node,$t36.node);})

    | (LEFTB AND t37=expr t38=exprplus RIGHTB {$node = new
lb_and_expr_exprplus_rb_expr($t37.node,$t38.node);} )
```

```
    | (LEFTB OR  t39=expr t40=exprplus RIGHTB {$node = new
lb_or_expr_exprplus_rb_expr($t39.node,$t40.node);})

    | (DISPLAY t41=name {$node = new display_name_expr($t41.node);})

    | (DISPLAY t42=string {$node = new display_string_expr($t42.node);})

    | NEWLINE  { $node= new newline_expr($NEWLINE.text);}

    | EMPTY { $node= new newline_expr($EMPTY.text);}

    | QUOTESX { $node= new newline_expr($QUOTESX.text);}

    | QSMARK { $node= new newline_expr($QSMARK.text);}

    | CHARACTERQUOTED { $node= new newline_expr($CHARACTERQUOTED.text);}

    | t43=reloperators {$node = $t43.node}

    | t44=arthoperators {$node = $t44.node}

    | t45=name {$node = $t45.node}

    | t46=number {$node = $t46.node}

    | t47=symbol {$node = $t47.node}

    | t48=bool {$node = $t48.node}

    | t49=string {$node = $t49.node}

    | t50=character {$node = $t50.node}
    ;

pkg  returns [ASTNode node] : (LEFTB t1=string t2=string t3=number t4=number
RIGHTB {$node = new
lb_string_string_number_number_rb($t1.node,$t2.node,$t3.node,$t4.node);});

name  returns [ASTNode node]:(NAME { $node= new name_nameplus($NAME.text);} );

symbol returns [ASTNode node]: (SYMBOL { $node= new
symbol_expr($SYMBOL.text);} );

number  returns [ASTNode node]: (INT { $node= new int_expr($INT.text);})
                                 | (DECIMAL { $node= new
decimal_expr($DECIMAL.text);});
```

```
bool  returns [ASTNode node]: (BOOLEAN { $node= new
boolean_expr($BOOLEAN.text);});

string  returns [ASTNode node] : (STRING { $node= new
string_expr($STRING.text);});

character  returns [ASTNode node]: (CHARACTER { $node= new
character_expr($CHARACTER.text);}) ;

reloperators  returns [ASTNode node]: (RELOPERATORS { $node= new
reloperators_expr($RELOPERATORS.text);});

arthoperators  returns [ASTNode node]: (ARTHOPERATORS { $node= new
arthoperators_expr($ARTHOPERATORS.text);});

BEGINN : 'begin';
BEGINN0 : 'begin0';
SETNQ : 'set!';
SET : 'set';
DELAY : 'delay';
CAR : 'car';
CDR : 'cdr';
COMBINATIONS : 'combinations';
LIST : 'list';
REVERSE : 'reverse';
APPEND : 'append';
LAMBDA : 'lambda';
LAMBDASYM : 'λ';
LOCAL : 'local';
LETREC : 'letrec';
SHARED : 'shared';
LET : 'let';
LETSTAR : 'let*';
RECUR : 'recur';
COND : 'cond';
ELSE : 'else ';
IF : 'if';
AND : 'and';
OR : 'or';
TSCHECKEXP : 'check-expect';
TSCHECKRAND : 'check-random' ;
TSCHECKWITHIN : 'check-within' ;
TSCHECKMEMBEROF :'check-member-of' ;
TSCHECKSATSIS : 'check-satisfied';
TSCHECKERROR :'check-error' ;
REQUIRE : 'require';
```

```
DISPLAY : 'display';
DEFINE : 'define';
NEWLINE : 'newline';
EMPTY : 'empty';
DEFINESTRUCT : 'define-struct';
QSMARK : '?';
QUOTESX: ''()';
QUOTEQUOTED : ''';
LEFTB : '(';
RIGHTB : ')';
LEFTSQB : '[';
RIGHTSQB : ']';
QUOTEQUASIQUOTED : '`' ;
CHARACTERQUOTED : '\u0027' '()';
ARTHOPERATORS : '+'
ARTHOPERATORS : '-'
ARTHOPERATORS : '*'
ARTHOPERATORS : '/';
BOOLEAN : '#true'
BOOLEAN : '#T'
BOOLEAN : '#t'
BOOLEAN :'#false'
BOOLEAN : '#F'
BOOLEAN : '#f';
RELOPERATORS :'<'
RELOPERATORS :'='
RELOPERATORS :'>';
SYMBOL : ([$%&!*-+\\^_~])+;
INT: [1-9] [0-9]*
INT: '0';
DECIMAL : INT '.' [0-9]+;
NAME: ([--:A-Za-z])+;
COMMA : ',';
COMMAAT : ',@';
STRING: '"' ([ -~])* '"';
CHARACTER : '#' '\u005C' [A-Za-z0-9]
CHARACTER : '#' '\u005C' 'space';
LANG: '#lang' ~ ('\n' | '\r')* '\r'? '\n' -> skip;
COMMENT: ';' ~ ('\n' | '\r')* '\r'? '\n' -> skip;
WS: (' ' | '\r' | '\t' | '\u000C' | '\n') -> skip;
```