CSE2260 - Principles of Programming Languages

PROJECT 1 – Part 2 (Due 23/05/2023 23:59)

In this project, you will implement a syntax analyzer (parser) for the given specific programming language, i.e., PPLL. The details of the PPLL are given below. This assignment focuses on implementing a recursive-descent parser (syntactic analyzer) for PPLL.

In this part of the Project, you will use the tokens produced by the scanner you implemented in the first part. The grammar for a syntactically correct PPLL program is given below:

<Program> → ε | <TopLevelForm> <Program>

<TopLevelForm> → **(** <SecondLevelForm> **)**

<SecondLevelForm> → <Definition> | **(** <FunCall> **)**

<Definition> → **DEFINE** <DefinitionRight>

<DefinitionRight> → **IDENTIFIER** <Expression> | **(IDENTIFIER** <ArgList> **)** <Statements>

<ArgList> → ε | **IDENTIFIER** <ArgList>

<Statements> → <Expression> | <Definition> <Statements>

<Expressions> → ε | <Expression> <Expressions>

<Expression> → **IDENTIFIER** | **NUMBER** | **CHAR** | **BOOLEAN** | **STRING** | **(** <Expr> **)**

<Expr> → <LetExpression> | <CondExpression> | <IfExpression> | <BeginExpression> | <FunCall>

<FunCall> → **IDENTIFIER** <Expressions>

<LetExpression> → **LET** <LetExpr>

<LetExpr> → **(** <VarDefs> **)** <Statements>

| **IDENTIFIER (** <VarDefs> **)** <Statements>

<VarDefs> → **(IDENTIFIER** <Expression> **)** <VarDef>

<VarDef> → ε | <VarDefs>

<CondExpression> → **COND** <CondBranches>

<CondBranches> → **(** <Expression> <Statements> **)** <CondBranches>

<CondBranch> → ε | **(** <Expression> <Statements> **)**

<IfExpression> → **IF** <Expression> <Expression> <EndExpression>

<EndExpression> → ε | <Expression>

<BeginExpression> → **BEGIN** <Statements>

The tokens in the grammar are written in **bold.** The parentheses are not written as their token names to make it simple (yes, only the **LEFTPAR** and **RIGHTPAR** are included in the grammar). The parenthesis must match.

Your task is to implement a recursive-descent parser that takes a PPLL program and prints its parse tree to stdout if the input is in fact a syntactically correct program. If the input is lexically incorrect, the parser should output the scanner's error message as in first part of the project. If the input is lexically correct but syntactically incorrect, the parser should output a syntax error for the first invalid token. Here are two examples:

Correct input: On the following input and using the above grammar, the parse tree your parser should print is shown in the next page. (The __ lines represent ε's in the above grammar.)

```
(define (fibonacci n)
      ( let fib ((prev 0) (cur 1) (i 0))
            ( if (= i n) cur (fib cur (+ prev cur) (+ i 1)))))
```

Incorrect input: Lexically incorrect inputs should be treated exactly as in first phase of the project.

The following is an input that is lexically correct (all its components are valid tokens), but it is syntactically incorrect because the last statement in the let-block is not an expression:

```
(define (fibonacci n)
       ( let fib ((prev 0) (cur 1) (i 0))
             ( if (= i n) cur (fib cur (+ prev cur) (+ i 1)))
             (define junk n)))
```

In this example your parser will give an error when it encounters the LEFTPAR token which is given in red in the expression. The output of your parser will be the parse tree up to the error and then a corresponding error message. It will output the type of the syntax error along with the line and column numbers of the error. An example output is given in the last page.


<Program>
<TopLevelForm>
LEFTPAR (()
<SecondLevelForm>
<Definition>
DEFINE (define)
<DefinitionRight>
LEFTPAR (()
IDENTIFIER (fibonacci)
<ArgList>
IDENTIFIER (n)
<ArgList>
—
RIGHTPAR ())
<Statements>
<Expression>
LEFTPAR (()
<Expr>
<LetExpression>
LET (let)
<LetExpr>
IDENTIFIER (fib)
LEFTPAR (()
<VarDefs>
LEFTPAR (()
IDENTIFIER (prev)
<Expression>
NUMBER (0)
RIGHTPAR ())
<VarDef>
<VarDefs>
LEFTPAR (()
IDENTIFIER (cur)
<Expression>
NUMBER (1)
RIGHTPAR ())
<VarDef>
<VarDefs>
LEFTPAR (()
IDENTIFIER (i)
<Expression>
NUMBER (0)
RIGHTPAR ())
<VarDef>
—
RIGHTPAR ())
<Statements>
<Expression>
LEFTPAR (()
<Expr>
<IfExpression>
IF (if)
<Expression>
LEFTPAR (()
<Expr>
<FunCall>
IDENTIFIER (=)
<Expressions>
<Expression>
IDENTIFIER (i)
<Expressions>
<Expression>
IDENTIFIER (n)
<Expressions>
—
LEFTPAR (()
<Expression>
IDENTIFIER (cur)
<EndExpression>
<Expression>
LEFTPAR (()
<Expr>
<FunCall>
IDENTIFIER (fib)
<Expressions>
<Expression>
IDENTIFIER (cur)
<Expressions>
<Expression>
LEFTPAR (()
<Expr>
<FunCall>
IDENTIFIER (+)
<Expressions>
<Expression>
IDENTIFIER (prev)
<Expressions>
<Expression>
IDENTIFIER (cur)
<Expressions>
—
RIGHTPAR ())
<Expressions>
<Expression>
LEFTPAR (()
<Expr>
<FunCall>
IDENTIFIER (+)
<Expressions>
<Expression>
IDENTIFIER (i)
<Expressions>
<Expression>
NUMBER (1)
<Expressions>
—
RIGHTPAR ())
<Expressions>
—
RIGHTPAR ())
RIGHTPAR ())
RIGHTPAR ())
RIGHTPAR ())
<Program>
—

```
<Program>
      <TopLevelForm>
            LEFTPAR (()
            <SecondLevelForm>
                  <Definition>
                        DEFINE (define)
                        <DefinitionRight>
                              LEFTPAR (()
                              IDENTIFIER (fibonacci)
                              <ArgList>
                                    IDENTIFIER (n)
                                    <ArgList>
                                          __
                              RIGHTPAR ())
                              <Statements>
                                    <Expression>
                                          LEFTPAR (()
                                          <Expr>
                                                <LetExpression>
                                                      LET (let)
                                                      <LetExpr>
                                                            IDENTIFIER (fib)
                                                            LEFTPAR (()
                                                            <VarDefs>
                                                                  LEFTPAR (()
                                                                  IDENTIFIER (prev)
                                                                  <Expression>
                                                                        NUMBER (0)
                                                                  RIGHTPAR ())
                                                                  <VarDef>
                                                                        <VarDefs>
                                                                              LEFTPAR (()
                                                                              IDENTIFIER (cur)
                                                                              <Expression>
                                                                                    NUMBER (1)
                                                                              RIGHTPAR ())
                                                                              <VarDef>
                                                                                    <VarDefs>
                                                                                          LEFTPAR (()
                                                                                          IDENTIFIER (i)
                                                                                          <Expression>
                                                                                                NUMBER (0)
                                                                                          RIGHTPAR ())
                                                                                          <VarDef>
                                                                                                __
                                                            RIGHTPAR ())
                                                            <Statements>
                                                                  <Expression>
                                                                        LEFTPAR (()
                                                                        <Expr>
                                                                              <IfExpression>
                                                                                    IF (if)
                                                                                    <Expression>
                                                                                          LEFTPAR (()
                                                                                          <Expr>
                                                                                                <FunCall>
                                                                                                      IDENTIFIER (=)
                                                                                                      <Expressions>
                                                                                                            <Expression>
                                                                                                                  IDENTIFIER (i)
                                                                                                            <Expressions>
                                                                                                                  <Expression>
                                                                                                                        IDENTIFIER (n)
                                                                                                                  <Expressions>
                                                                                                                        __
                                                                                          RIGHTPAR ())
                                                                                    <Expression>
                                                                                          IDENTIFIER (cur)
                                                                                    <EndExpression>
                                                                                          <Expression>
                                                                                                LEFTPAR (()
                                                                                                <Expr>
                                                                                                      <FunCall>
                                                                                                            IDENTIFIER (fib)
                                                                                                            <Expressions>
                                                                                                                  <Expression>
                                                                                                                        IDENTIFIER (cur)
                                                                                                                  <Expressions>
                                                                                                                        <Expression>
                                                                                                                              LEFTPAR (()
                                                                                                                              <Expr>
                                                                                                                                    <FunCall>
                                                                                                                                          IDENTIFIER (+)
                                                                                                                                          <Expressions>
                                                                                                                                                <Expression>
                                                                                                                                                      IDENTIFIER (prev)
                                                                                                                                                <Expressions>
                                                                                                                                                      <Expression>
                                                                                                                                                            IDENTIFIER (cur)
                                                                                                                                                      <Expressions>
                                                                                                                                                            __
                                                                                                                              RIGHTPAR ())
                                                                                                                        <Expressions>
                                                                                                                              <Expression>
                                                                                                                                    LEFTPAR (()
                                                                                                                                    <Expr>
                                                                                                                                          <FunCall>
                                                                                                                                                IDENTIFIER (+)
                                                                                                                                                <Expressions>
                                                                                                                                                      <Expression>
                                                                                                                                                            IDENTIFIER (i)
                                                                                                                                                      <Expressions>
                                                                                                                                                            <Expression>
                                                                                                                                                                  NUMBER (1)
                                                                                                                                                            <Expressions>
                                                                                                                                                                  __
                                                                                                                                    RIGHTPAR ())
                                                                                                                              <Expressions>
                                                                                                                                    __
                                                                                          RIGHTPAR ())
                                                                                    RIGHTPAR ())
SYNTAX ERROR [3:28]: ')' is expected
```