Sabancı University
Faculty of Engineering and Natural Sciences

CS305 Programming Languages

Homework 3

Due: 16-04-2018

1 Introduction

In this homework you will implement a semantic checker and translator for PLScript programming language which you have implemented a parser in the last assignment. A bison/flex implementation are provided to you. However the implementation of attribute grammar is missing.

A context free grammar given in Section 2. Please note that the grammar is slightly different than the grammar of the second homework.

The semantic rules that you have to check are listed below. Here is a semantically incorrect PLScript program which will be used to explain semantic errors:

```
 1  for(var x in z) {
 2    a = x + 1 + 2;
 3    foo();
 4  }
 5  function bar() {}
 6  for(t in []){
 7    function hello(a)  {
 8      a = t + 'hello' + 'world';
 9    }
10    bar();
11    hello();
12  }
13  hello();
14  function hello() {}
15  15 * 30;
16  1 + 2 + 3;
17  a = 3 - 2 - 1;
18  a + 10;
```

Figure 1: Example of a PLScript program

2 Grammar

In this section, we give the context free grammar that you will use.

prog	->	***statementList***
statementList	->	***statementList statement***
	|	***statementList*** tSEMICOLON
	|	ϵ
statement	->	***assign*** | ***if*** | ***expr while*** | ***for***
	|	***functionCall*** | ***functionDeclaration***
assign	->	tIDENT tEQ ***expr***
	|	tVAR tIDENT tEQ ***expr***
if	->	***ifPart elsePart***
ifPart	->	tIF tLPAR ***expr*** tRPAR ***statementBlock***
elsePart	->	tELSE ***statementBlock***
	|	ϵ
while	->	tWHILE tLPAR ***expr*** tRPAR ***statementBlock***
for	->	tFOR tLPAR tIDENT tIN ***expr*** tRPAR ***statementBlock***
	|	tFOR tLPAR tVAR tIDENT tIN ***expr*** tRPAR ***statementBlock***
functionDeclaration	->	tFUNCTION tIDENT tLPAR ***identList*** tRPAR ***statementBlock***
	|	tFUNCTION tIDENT tLPAR tRPAR ***statementBlock***
statementBlock	->	tLBRACE ***statementList*** tRBRACE
functionCall	->	tIDENT tLPAR ***exprList*** tRPAR
	|	tIDENT tLPAR tRPAR
expr	->	tIDENT | tREAL | tINT | tSTRING
	|	tLBRKT tRBRKT | tLBRKT ***exprList*** tRBRKT
	|	tLBRACE tRBRACE | tLBRACE ***propertyList*** tRBRACE
	|	tNOT ***expr*** | ***expr*** tPLUS ***expr*** | ***expr*** tMINUS ***expr***
	|	***expr*** tSTAR ***expr*** | ***expr*** tEQCHECK ***expr***
	|	***expr*** tLT ***expr*** | ***expr*** tGT ***expr***
exprList	->	***expr*** | ***exprList*** tCOMMA ***expr***
identList	->	tIDENT | ***identList*** tCOMMA tIDENT
propertyList	->	tIDENT tCOLON ***expr***
	|	***propertyList*** tCOMMA tIDENT tCOLON ***expr***

3 Semantic Rules

Semantic rules for PLScript:

SR1: **called function shall be declared before**

In PLScript language, when a function is called, the function shall be declared before in current scope or in the parent scope. Otherwise, an error message should be printed.

Examples:

(a) For the third line in Figure 1 the output will be as follows:
`ERROR: The function (foo) is not declared before`

(b) There is no error for the line ten in Figure 1 since the function bar is declared in parent scope.

(c) There is no error for the line eleven in Figure 1 since the function hello is declared in same scope.

(d) For the line thirteen in Figure 1 the output will be as follows:
`ERROR: The function (hello) is not declared before`

4 Translation

As the translation part, the program shall compute the value of an expression with following rules:

- ***expr*** `->` ***expr*** `tPLUS` ***expr***

  If the values of the expressions at right hand side are integers or real numbers, then the value of left hand side is sum of others. Alternatively if the values at the right hand side are strings then the resulting value is concatenation of these values.

- ***expr*** `->` ***expr*** `tMINUS` ***expr***

  If the values of the expressions at right hand side are integer or real numbers, then the value of left hand side is difference of others.

- ***expr*** `->` ***expr*** `tSTAR` ***expr***

  If the values of the expressions at right hand side are integer or real numbers, then the value of left hand side is product of others.

The program should print operation with infix notation then an arrow and the result.

Examples:

- For the second line in Figure 1, output will be as follows:
  `1 + 2 => 3`
- For the line eight in Figure 1, output will be as follows:
  `hello + world => helloworld`
- For the line fifteen in Figure 1, output will be as follows:
  `15 * 30 => 450`
- For the line sixteen in Figure 1, outputs will be as follows:
  `1 + 2 => 3`
  `3 + 3 => 6`

- For the line seventeen in Figure 1, outputs will be as follows:
  `3 - 2 => 1`
  `1 - 1 => 0`

- For the line eighteen in Figure 1, there will be no output since a is an variable.

5 Output

The program must print the outputs to the console. The program print the all errors and translation output if there any.

6 How to Submit

You need to design the semantic checker and the translator for PLScript with bison/flex files provided to you. You can also use additional header files (`.h` files) which are `#include`d from your files. Please also use a file named `parser_utils.c`, where you implement additional C functions that are used by your program.

Zip your files named as `id-hw3.zip` where id is your student ID. Please do not change file names. We will compile your files by using the following commands:

```
flex scanner.flx
bison -d parser.y
gcc -c parser_utils.c
gcc -o parser parser_utils.o lex.yy.c parser.tab.c -lfl
```

So, make sure that these four commands are enough to produce the executable semantic checker and translator. If we assume that there is a text file named `test` having a PLScript program, we will execute your semantic checker by using the following command line:

```
parser < test
```

The output should be displayed on the screen.

7 Notes

- **Important:** SUCourse's clock may be off a couple of minutes. Take this into account to decide when to submit.

- No homework will be accepted if it is not submitted using SUCourse.

- You must write your files by yourself.
- Start working on the homework immediately.