Compilers

CMPT 432

Project Three - 150 points

Project

1. Projects one and two working perfectly. [-∞ if not]
2. Create an AST from the tokens or from the CST. Display it. [50 points]
3. Write a semantic analyzer that scope-checks and type-checks the AST based on the grammar found on our class web site at http://www.labouseur.com/courses/compilers/grammar.pdf and the type rules we discussed in class. [75 points]
4. Create and display a symbol table with type and scope information. [25 points]

Notes and Requirements

- Build an Abstract Syntax Tree either from the tokens or by modifying the CST. Display it after successful lex and parse.
- Scope-check on the AST.
- While you are scope-checking (or afterwards), build a symbol table (perhaps using the CST or AST) of IDs that includes their name, data type, scope, position in the source code, and anything else you think might be important.
- Type-check the source code using the AST and the symbol table.
  - Issue errors for undeclared identifiers, redeclared identifiers in the same scope, type mismatches, and anything else that might go wrong.
  - Issue warnings about declared but unused identifiers, use of uninitialized variables, and use of initialized but unused variables.
- Include verbose output functionality that traces the semantic analysis stages, including scope checking, the construction of the symbol table, and type checking actions.
- When you detect an error, report it in helpful detail including where it was found
- See examples on next several pages for details and ideas.

Other Requirements

Create several test programs that cause as many different types of errors as you can in order to thoroughly test your code. Include several test cases that show it working as well. Write up your testing results (informally) in a document in your Git repo.

Your code must … [−∞ if not]

- separate structure from presentation.
- be professionally formatted.
- use and demonstrate best practices.
- make me proud to be your teacher.

Labs

Labs 5, part of 6, and 7 focus on the components of this project.

Submitting Your Work

Make many commits to GitHub. I do not want to see one massive "everything" commit when I review your code. (It's −∞ if you do that.) Commit early and often.

E-mail me the URL to your private GitHub master repository. Remember to add me (Labouseur) as a collaborator. Please send this to me before the due date (see our syllabus).

Input file:

```
{
  int a
  boolean b
  {
    string c
    a = 5
    b = true  /* no comment */
    c = "inta"
    print(c)
  }
  print(b)
  print(a)
}$

{
  int a
  {
    boolean b
    a = 1
  }
  print(b)
}$
```

Output to screen:

```
Program 1 Lexical Analysis
Program 1 Lexical analysis produced 0 error(s) and 0 warning(s)

Program 1 Parsing
Program 1 Parsing produced 0 error(s) and 0 warnings

Program 1 Semantic Analysis
Program 1 Semantic Analysis produced 0 error(s) and 0 warning(s)

Program 1 Concrete Syntax Tree
----------------------------------------
< Program >
-< Block >
--[ { ]
--< Statement List >
---< Statement >
----< Variable Declaration >
-----[ int ]
-----[ a ]
---< Statement List >
----< Statement >
-----< Variable Declaration >
------[ boolean ]
------[ b ]
----< Statement List >
-----< Statement >
------< Block >
-------[ { ]
-------< Statement List >
```

--------< Statement >
----------< Variable Declaration >
----------[string]
----------[c]
--------< Statement List >
---------< Statement >
----------< Assignment Statement >
-----------[a]
-----------[=]
-----------< Expression >
------------< Int Expression >
-------------[5]
---------< Statement List >
----------< Statement >
-----------< Assignment Statement >
------------[b]
------------[=]
------------< Expression >
-------------< Boolean Expression >
--------------[true]
----------< Statement List >
-----------< Statement >
------------< Assignment Statement >
-------------[c]
-------------[=]
-------------< Expression >
--------------< String Expression >
---------------["]
---------------< Char List >
----------------[i]
----------------< Char List >
-----------------[n]
-----------------< Char List >
------------------[t]
------------------< Char List >
-------------------[a]
-------------------< Char List >
---------------["]
-----------< Statement List >
------------< Statement >
-------------< Print Statement >
--------------[print]
--------------[( ]
--------------< Expression >
---------------[c]
--------------[ )]
------------< Statement List >
-------[}]
-----< Statement List >
------< Statement >
-------< Print Statement >
--------[print]
--------[( ]
--------< Expression >
---------[b]
--------[ )]


```
------< Statement List >
-------< Statement >
--------< Print Statement >
---------[ print ]
---------[ ( ]
---------< Expression >
----------[ a ]
---------[ ) ]
-------< Statement List >
--[ } ]
-[ $ ]
```

Program 1 Abstract Syntax Tree

```
----------------------------------------
< BLOCK >
-< Variable Declaration >
--[ int ]
--[ a ]
-< Variable Declaration >
--[ boolean ]
--[ b ]
-< BLOCK >
--< Variable Declaration >
---[ string ]
---[ c ]
--< Assignment Statement >
---[ a ]
---[ 5 ]
--< Assignment Statement >
---[ b ]
---[ true ]
--< Assignment Statement >
---[ c ]
---[ inta ]
--< Print Statement >
---[ c ]
-< Print Statement >
--[ b ]
-< Print Statement >
--[ a ]
```

Program 1 Symbol Table

Name	Type	Scope	Line
a	int	0	2
b	bool	0	3
c	string	1	5

Program 2 Lexical Analysis
Program 2 Lexical analysis produced 0 error(s) and 0 warning(s)

Program 2 Parsing
Program 2 Parsing produced 0 error(s) and 0 warning(s)

Program 2 Semantic Analysis
Error: The id b on line 7 was used before being declared
Program 2 Semantic Analysis produced 1 error(s) and 0 warning(s)

Program 2 Concrete Syntax Tree

```
----------------------------------------
< Program >
-< Block >
--[ { ]
--< Statement List >
---< Statement >
----< Variable Declaration >
-----[ int ]
-----[ a ]
---< Statement List >
----< Statement >
-----< Block >
------[ { ]
------< Statement List >
-------< Statement >
--------< Variable Declaration >
---------[ boolean ]
---------[ b ]
-------< Statement List >
--------< Statement >
---------< Assignment Statement >
----------[ a ]
----------[ = ]
----------< Expression >
-----------< Int Expression >
------------[ 1 ]
--------< Statement List >
------[ } ]
----< Statement List >
-----< Statement >
------< Print Statement >
-------[ print ]
-------[ ( ]
-------< Expression >
--------[ b ]
-------[ ) ]
-----< Statement List >
--[ } ]
-[ $ ]
```

Program 2 Abstract Syntax Tree

```
----------------------------------------
< BLOCK >
-< Variable Declaration >
```



```
--[ int ]
--[ a ]
-< BLOCK >
--< Variable Declaration >
---[ boolean ]
---[ b ]
--< Assignment Statement >
---[ a ]
---[ 1 ]
-< Print Statement >
--[ b ]
```

Program 2 Symbol Table
not produced due to error(s) detected by semantic analysis

