Compilers

CMPT 432

Project Two - 100 points

Project

1. Project one working perfectly. [−∞ if not]
2. Write a recursive descent parser and add it to your compiler. The parser must validate the tokens you lexed from project one. [70 points]
3. While parsing, create a concrete syntax tree (CST). If parsing is successful (i.e., no errors were found) then display the CST. Make it neat and make it pretty. [30 points]

Notes and Requirements

- Your compiler must compile multiple programs in sequence. As with your lexer, each program must be separated by the $ [EOP] marker in the source code.
- Do not attempt any semantic analysis for this project. No type checking, no scope checking, no AST… save it for the next project.
- Provide both errors and warnings. Warnings are non-fatal mistakes or omissions.
- When you detect an error, report it in helpful and excruciating detail including where it was found, what exactly went wrong, and how the programmer might fix it.
- When there are errors detected in lex, do not continue to parse.
- When there are errors detected in parse, do not display the CST.
- Include verbose output functionality that traces the stages of the parser including the construction of the symbol table.
- See the examples on the next few pages for details and ideas.

Other Requirements

You have to write this yourself. You may not use JavaCC, ANTLR, or any compiler compiler.

Create many test programs that cause as many different types of errors as you can in order to thoroughly test your code. (Keep thinking about code coverage and edge cases). Include a bunch of 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 and still show your uniqueness
- use and demonstrate best practices.
- make me proud to be your teacher.

Hints

Remember the utility of comments and how much their presence and quality affect your professionalism and my opinion of your work.

Labs

Labs 3, 4, and 5 focus on the components of this project and the mid-term exam.

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. Another 40 - 60 commits is a good goal for this project.

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).

Compilers

CMPT 432

Input:

```
{}$
{{{{{{}}}}}}$
{{{{{{}}} /* comments are ignored */ }}}}$
{ /* comments are still ignored */ int @}$
```

Output to screen:

```
DEBUG: Running in verbose mode

LEXER: Lexing program 1...
LEXER: "{" --> [LBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "$" --> [EOP]
LEXER: Lex completed successfully

PARSER: Parsing program 1...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1...
<Program>
-<Block>
--[{]
--<Statement List>
--[}]
-[$]


LEXER: Lexing program 2...
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "$" --> [EOP]
LEXER: Lex completed successfully

PARSER: Parsing program 2...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 2...
<Program>
-<Block>
--[{]
--<Statement List>
---<Statement>
----<Block>
-----[{]
-----<Statement List>
------<Statement>
-------<Block>
--------[{]
--------<Statement List>
---------<Statement>
----------<Block>
-----------[{]
-----------<Statement List>
------------<Statement>
-------------<Block>
--------------[{]
--------------<Statement List>
---------------<Statement>
----------------<Block>
-----------------[{]
-----------------<Statement List>
-----------------[}]
--------------[}]
-----------[}]
--------[}]
-----[}]
--[}]
-[$]
```

```
LEXER: Lexing program 3...
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "{" --> [LBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "}" --> [RBRACE]
LEXER: "$" --> [EOL]
LEXER: Lex completed successfully

PARSER: Parsing program 3...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: ERROR: Expected [EOL] got [RBRACE] with value '}' on line 0
PARSER: Parse failed with 1 error

CST for program 3: Skipped due to PARSER error(s).


LEXER: Lexing program 4...
LEXER: "{" --> [LBRACE]
LEXER: "int" --> [TYPE]
LEXER: ERROR: Unrecognized Token: @
LEXER: "}" --> [RBRACE]
LEXER: "$" --> [EOL]
LEXER: Lex completed with 1 error

PARSER: Skipped due to LEXER error(s)

CST for program 4: Skipped due to LEXER error(s).
```