Concordia University
Department of Computer Science and Software Engineering

Compiler Design (COMP 442/6421)
Winter 2022

Assignment 3: Abstract Syntax Tree Generation

Deadline:	Sunday March 6th, 2022
Evaluation:	8% of final grade
Late submission:	penalty of 50% for each late working day

This assignment is about complementing the parser developed in assignment 2 with syntax-directed translation to generate an abstract syntax tree (AST) data structure that is going to be used in the later stages of the compiler as an intermediate representation. The parser must still include the features developed in assignment 2.

The assignment handout includes two example source files. These files should be used as-is and not be altered in any way. Completeness of testing is a major grading topic. You are responsible for providing appropriate test cases that test for a wide variety of valid cases in addition to what is in the example source files provided.

Work to submit

Document

You must provide a short document that includes the following sections:

Section 1. **Attribute grammar** : Provide an attribute grammar that clearly indicates where each of the semantic actions are inserted in the grammar's right hand sides, as well as a list of all the semantic actions and a brief description of their respective effect. See slide set 8.5 for an example.

Section 2. **Design** : Give a brief overview of the overall structure of your solution, as well as a brief description of the role of each component of your implementation.

Section 3. **Use of tools** : Identify all the tools/libraries/techniques that you have used in your analysis or implementation and justify why you have used these particular ones as opposed to others.

Implementation

- **Parser augmented with syntax-directed translation**: implementation of syntax-directed translation in the existing parser for the purpose of generating an AST data structure. The result of the execution of the parser should be the creation of an AST data structure that corresponds to the parse tree as identified by the parsing process. The tree must be an AST, and not a parse tree. This tree will become the intermediate representation used by the two following assignments.
- **AST output**: The generated tree should be output to a file in a format that allows easy visualization of the structure of the tree. This can be a simple text representation of the tree structure or a graphiz dot representation (see examples in the assignment handout). This file must allow the marker to quickly verify that the generated AST is in fact correct with regards to the parsed program. When parsing a file named, for example, **`originalfilename`**, the parser should write into a file named ***`originalfilename`*****`.outast`** a text representation of the abstract syntax tree representing the original program.
- **Test cases** : Provide a set of source files that enable to test the AST generation for all syntactical structures involved in the language.
- **Driver**: Include a driver that parses all your test files. For each test file, the corresponding **`outast`** files should be generated (in addition to all the files that were generated by the previous assignments).

Assignment submission requirements and procedure

- Each submitted assignment should contain four components: (1) the source code, (2) a group of test files, (3) a brief report, and (4) an executable named **`ASTdriver`**, that extracts the tokens from all your test files. For each test file, the corresponding **`outast`** files should be generated.
- The assignment statement provides test files (**`.src`**) and their corresponding output files.
- The source code should be separated into modules using a comprehensible coding style.
- The assignment should be submitted through moodle in a file named: "**`A#_student-id`**" (e.g. **`A1_1234567`**, for Assignment #1, student ID 1234567) and the report must in a PDF format.
- You may use any language you want in the project and assignments but the only fully supported language during the lab is Java.
- You have to submit your assignment before midnight on the due date on moodle.
- The file submitted must be a **.zip** file

The marking will be done in a short presentation to the marker. A schedule will be provided to you by email in the days before the due date. You will be given a short time for the presentation, so make sure that you are ready to effectively demonstrate all the elements mentioned in the "Work to submit" section above.

Evaluation criteria and grading scheme

Analysis:		
Attribute grammar that clearly indicates where each of the semantic actions are inserted in the grammar's right hand sides, as well as a list of all the semantic actions and a brief description of their respective effect. See slide set 8.5 for an example – document Section 1.	ind 2.1	5 pts
Design/implementation:		
Description/rationale of the overall structure of the solution and the roles of the individual components used in the applied solution – document Section 2.	ind 4.3	5 pts
Correct implementation of syntax-directed translation implemented as part of a top-down predictive parser following the grammar given in this handout, which is generating and AST (not a parse tree) data structure that correctly represents any valid program given in input.	ind 4.4	20 pts
Output of the AST in an **`outast`** file.	ind 4.4	5 pts
Completeness of test cases.	ind 4.4	10 pts
Use of tools:		
Description of tools/libraries/techniques used in the analysis/implementation. Description of other tools that might have been used. Justification of why the chosen tools were selected – document Section 3.	ind 5.2	2 pts
Successful/correct use of tools/libraries/techniques used in the analysis/implementation.	ind 5.1	3 pts
Total		**50 pts**