Assignment #3

Date Due: March 31, 2026

Total: 100 marks

Instructions

- Submit all program code and any relevant output from program testing.
- Combine all your files into a single compressed file. Please use a filename which includes your UPEI username and is of the form: username_ast3.tar.gz; submit the targzip file via moodle.

  Each program includes beside the source code an executable Readme file which will run all examples automatically and a run.txt file containing the demo of the execution, for each test.

Requirements for building a parser

1. Some explanations may be submitted on paper, i.e., txt/pdf (the expanded grammar, scanner construction etc).

   You need to create some grammars that will allow to write a direct syntax translation parser, an $LL(1)$, or a precedence grammar one for a simple programming language (that will include constructions for assignment expressions, decision statements, and loops).

   The action routines in the parser should write to standard output the production being used to make a reduction. That is, your task is to produce a derivation for any valid program.

   (a) Part 1 (15 marks)

   Create a grammar for a sequence of expressions; use a delimiter such as a comma or semicolon to separate them. The expressions match exactly those from your programming language. Expressions should permit the use of arithmetic, logical and relational operators. Assignments could appear in expressions as in C or in statements as in Pascal.

   Do not attempt to do type checking in this grammar (it is not required).

   Sample input could be given as:

   $\texttt{55.33}; \texttt{x} + \texttt{y} * (\texttt{234} - \texttt{x}); \texttt{myIDnumber} \geq \texttt{999}; \texttt{x} : \texttt{2} < (\texttt{5} * (\texttt{w} + \texttt{t});$

(b) Part 2 (15 marks)

Create a separate grammar for a sequence of statements. In this grammar, "expression" should be treated as a terminal or token.

(c) Part 3 (15 marks)

Create a grammar for the remainder of your language. The grammar rules for declarations should be part of this grammar. "Statement" should be treated as a token.

This grammar represents the entire contents of a compilation file. For example, in C, it would be a sequence of external declarations and definitions of variables and functions.

(d) Part 4 (30 marks)

Combine the grammars from 1a, 1b, and 1c. This gives a grammar for a complete program file in your language.

(e) Testing and Verification of Program Correctness (25 marks)

Demonstrate the parsing of valid programs for the grammars above, and as well, show how the parser responds to syntax errors.

(f) (20 marks) Design a number of test cases to provide reasonable confidence that your translation scheme is correct.

Run your test cases and produce a file for the output of each test (record the execution in a file). Provide an overall summary of your test runs and the confidence you have that it is working correctly.