Formal Languages and Logic, COMP2022
Assignment 2

DUE Friday 8 May 3 pm

Consider the following grammar G, with start variable S:

S	→	E | S A | A
E	→	if (C) {S} | if (C) {S} else {S}
A	→	V := T ;
T	→	a | b
V	→	x | y
C	→	V O T | T O V | V O V | T O T
O	→	< | >

1) a) Give its set of variables and its set of terminals.
   b) Give the parse tree for:
   "if (x > y) { if (y < a) {x := a; y :=b;} else {x := b;}}"
2) Show that this grammar is not LL(1) as is. Modify the rules appropriately to obtain an equivalent grammar G' that could be LL(1).
3) Calculate the FIRST sets for all the rules of G'. Only when it is necessary (make sure you only calculate the necessary ones), build the appropriate FOLLOW sets. Confirm whether or not G' is LL(1).
4) Build the LL(1) parse table for G'. Explain how you build this table and show it.
5) Implement the LL(1) table-driven parser in the language of your choice (see below the available options). Your program should take a filename as a command line argument, and do the following on the contents of the file:
   (i) Output the left most derivation steps with the stack contents for each step (one step per line on the screen)
   (ii) Output "ACCEPTED" or "REJECTED", depending on whether the contents of the file are accepted or rejected by the LL(1) parser
   (iii) Output appropriate error messages if the input was rejected. For instance "expected a '}' instead of 'x'"

   Note: all whitespace in the input file should be ignored (line breaks, spaces, etc.)
6) Explain what would be required in your program if we wanted to add or modify a rule (you do not need to change your program, just point to the changes required)
7) Implement an error recovery feature, where your program tries and recovers from mistakes and continue its parsing after issuing a warning. Explain how you implemented this feature. This feature should only run if a second command line argument "-e" is provided, after the input filename. If only one command line argument is used, error recovery must not be performed.
8) Provide testing files and a test summary, which should explain what you tested for, whether the test passed, with expected and observed output. These will be marked for their relevance and coverage. As a start, you are provided with 2 test files, accept.txt and error_missing_semicolon.txt which should pass and fail respectively.

Submission deadline: Friday 8 May, week 9, 3pm.

Late policy submission: A penalty of -2 marks will apply for each day late and the assignment will not be accepted if it is submitted more than 7 days after the due date .

Submission instructions: You need to submit both:

1) Your code in PASTA.

2) Your report in elearning.

Programming language: You can write the program in Java, C, C++, or Python.

Weight: This assignment is worth 16% of your final mark

Marking scheme

Marking scheme (total 16 marks):

1) 1 mark (0.5+0.5)
2) 2.5 marks
3) 2 marks
4) 1.5 mark
5) 4 marks:
    - 1 mark for the quality of the code (comments, choice of classes)
    - 2 marks for correctness
    - 1 mark for error messages
6) 1 mark for genericity of your program
7) 2.5 marks
8) 1.5 mark

Penalty: up to -1 mark for poorly presented report, absent or unclear explanations, ill-commented program.