Compilation - Final Exam

(**Duration:** 02 hours - **No electronic devices or documents permitted**)

Exercise 1 (6 *pts*)

Consider the grammar G with $V = \{A, B, C\}$, $T = \{or, and, not, (,), true, false\}$, $S = A$ and P :

A → A or B | B
B → B and C | C
C → not C | (A) | true | false

1. Eliminate left recursion from G and give the resulting equivalent grammar G'.
2. Construct the LL(1) parsing table for G'. Is the grammar G' LL(1)? Justify.
3. Write the corresponding ***Bison*** code for the grammar G.

Exercise 2 (7 *pts*)

Consider the following grammar:

S → A | B
A → i A e A | a
B → i A e B | i S

1. Construct the canonical collection of LR(0) items and draw the GOTO graph.
2. Construct the SLR(1) parsing table.
3. Determine whether this grammar is SLR(1) and justify your answer.
4. Using your parsing table, show the step-by-step parsing of the input string: ***i i a e a***

Exercise 3 (7 *pts*)

Consider a mini-language for robot movement commands, which includes commands like:

```
MOVE +10
TURN LEFT
REPEAT 3 { MOVE 5 TURN RIGHT }
STOP
```

The language has three main types of tokens:

- ***Keywords***: **`MOVE`** (moves the robot a distance), **`TURN`** (turns the robot LEFT or RIGHT), **`LEFT`**, **`RIGHT`** , **`REPEAT`** (repeats a block of commands a fixed number of times), and **`STOP`**.
- ***Numbers***: integers representing distances or repeat counts
  - **For MOVE command**: non-zero integer without leading zeros, optionally preceded by + or -
  - **For REPEAT command**: non-zero positive integer ≤ 100 (without leading zeros)
- ***Braces /Separators***: {, } and whitespace (spaces, tabs, newlines). Whitespace should be ignored.

1. Write lexical specification for all token types using regular expressions.
2. Draw the Deterministic Finite Automaton (DFA) that recognizes all tokens.
3. Write a ***Flex*** program that recognizes all tokens of the robot movement language and, at the end, prints the total distance walked by the robot. An example of input for the lexer is given below:

Example Input:	***Expected Output:***
MOVE +10	**Token:** MOVE, **Lexeme:** MOVE
TURN LEFT	**Token:** NUMBER, **Lexeme:** +10
MOVE -5	**Token:** TURN, **Lexeme:** TURN
STOP	**Token:** LEFT, **Lexeme:** LEFT
	Token: MOVE, **Lexeme:** MOVE
	Token: NUMBER, **Lexeme:** -5
	Token: STOP, **Lexeme:** STOP
	Total distance walked: 5