UNIVERSITY COMPUTER SCIENCE • EXERCISE SHEET

11.4 Lexical analysis

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS tokens • deterministic finite automaton

Objective

Lex by hand with longest match, positions and policy.

1 Key knowledge and worked examples

■ **The pipeline**

Regexes compile to DFAs; the lexer emits (type, lexeme, position) with greedy longest match and priority tie-breaks ✓.

■ **Worked example: the course line**

`count = a1 + b2;` → IDENT ASSIGN IDENT PLUS IDENT SEMI at columns 1, 7, 9, 12, 14, 16 ✓. `ifx` is one IDENT ✓.

2 Practice

2.1 Why must the lexer keep positions? [3]

2.2 What wins between `if` and `ident` on the input `ifx`? [3]

2.3 What is discarded on the way through? [3]

3 Exam-style questions

3.1 Lex `total = ifx + 42;` under the course specification (keyword `if`; IDENT letters/digits; NUM digits; operators =, +; punctuation ;).

List the token stream with type, lexeme and column for each. Justify the `ifx` decision by longest-match-plus-priority. State the NUM policy for a following letter (`42abc`) under our spec. State which single token would change if keywords were removed from the spec. [8]

3.2 Two error policies for `12abc`: report as an error, or emit NUM(12) then IDENT(abc).

State what each policy makes legal downstream. Argue which fits a language where identifiers cannot start with a digit. State the general principle the choice illustrates about lexer specification ✓. [7]