Course Administration

- Exercise 1 Redo
- Exercise 2 Preliminary Exercise
- Project 1 Prelim will be assigned on Wednesday
Regular Expressions

- Alphabet – the symbols that actually appear in the lexeme
- Special symbols to define the regular expression
  - ( ) : grouping
  - * : 0 or more occurrences of a pattern
  - + : 1 or more occurrences of a pattern
  - | : indicates alternatives
  - λ : indicates nothing (lambda)
Regular Expression Examples

- Alphabet = \{a,b,c\}
- Examples
  - a (b | c) a →
  - a\(^+\) (b | c) a\(^+\) →
  - a (b | c)* a →
  - abc*ba →
  - (a|b|c|\(\lambda\))(ab*\(c\)|(cb*a))\(^+\) →
Regular Expression for User Defined Names

- Alphabet = {_, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

- Regular expression?
  - (_|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z)(_|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|0|1|2|3|4|5|6|7|8|9)*
A regular expression for unsigned integer numeric literals

- Alphabet = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}
- Regular expression?
  - \((0|1|2|3|4|5|6|7|8|9)^+ \leq 2^{32} \text{ or } 2^{64}\)
A regular expression for signed integer numeric literals

- Alphabet = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, +\}
- Regular expression?
  - (+|- |λ) (0|1|2|3|4|5|6|7|8|9)^+ <= 2^{-32} to 2^{32} – 1
  - +55   -55   55
  - What about -000000000?
Deterministic Finite Automata

- States
  - 5

- Transitions
  - a

- Start state
  - 1

- Final state
  - 9
DFAs for Examples

- $a \ (b \ | \ c) \ a$
- $a^+ \ (b \ | \ c) \ a^+$
- $a \ (b \ | \ c)^* \ a$
- $abc^*ba$
DFAs for Examples

- $a \ (b \mid c) \ a$
  - aba
  - abc
- $a^+ \ (b \mid c) \ a^+$
  - aaabaaaaa
  - aaabaacaa
- $a \ (b \mid c)^* \ a$
  - acba
  - aaa
- $abc^*ba$
  - abba
  - acbba
DFAs as scanners (aka tokenizers)

- Alphabet = \{a, b, c, x, y, z, \sim\}

- Regular expression 1 (RE1)
  - a* (ab | bc) a+
  - DFA

- Regular expression 2 (RE2)
  - x+ (xy | yz | xz) z*
  - DFA
Adding terminating states

- Successful states
- Error states
Programming a DFA

- Table