COSC 445 **Project #1** **Lexical Analysis** **Distributed:** 14 Jan 2010
Due: 28 Jan 2010

See text pp 398-9 for specification of tokens in Triangle.

Synopsis: Give DFAs, state transition tables, working code (single program) and a demonstration of a tokenizer for the portion of the Triangle language.

Longer statement: Implement a tokenizer for the following tokens in Triangle:

- Integer-Literal
- Character-Literal
- Identifier
- Operator
- ~~Blank~~

Supply the DFA and the state transition table for each (5 DFAs).
Implement the DFAs (table-driven or direct-coded or other) in a single program.

Emit each token (type and value) as it is recognized. Each token is represented as an object (or struct); the format of the output is up to you.

The tokenizer should consume as many characters as possible (e.g., `'abc'` is recognized as the single token `[Identifier, 'abc']`, not as the three tokens `[Identifier, 'a'], [Identifier, 'b'], [Identifier, 'c']`)

End the program, with appropriate message, when an error is detected or when the end of the input character stream is reached.

Test cases: Supply your test input streams in the following format (spreadsheet format). I'm giving a few examples to clarify. There is no implication that a given condition will have exactly one test case.

Condition	*Test case*
Successful Identifier-only stream	x y value in var123
Identifier-failure	x&a x y

There must be a minimum of 10 test cases. Your grade will include how well your test cases cover all possible things that can go wrong.

Turn in:

- Hardcopy of program
- DFAs and state transition tables
- List of test cases
- Demo of program on your test cases (hard copy or live)

Grade based on:

- Program (65%)
  - Satisfies specification
  - Elegance and readability (see Style Standards)
  - Demo
- DFAs and Transition tables (10%)
- Test cases (25%)