Project 1 Worksheet

By answering the following questions, show that you know how the formal definition of a finite state machine and of a finite state transducer corresponds to the given code for Project 1.

A *deterministic finite state machine* is a quintuple (Σ,*S*,*s*0,δ,*F*), where:

* + Σ is the input alphabet (a finite, non-empty set of symbols).
  + *S* is a finite, non-empty set of states.
  + *s*0 is an initial state, an element of *S*.
  + δ is the state-transition function: δ : *S* ×Σ → *S*.
  + *F* is the set of final states, a (possibly empty) subset of *S*.

A *finite state transducer* is a 6-tuple (Σ,Γ,*S*,*s*0,δ,*F*) as above except:\

* + Γ is the output alphabet (a finite, non-empty set of symbols)
  + δ is the state-transition function: δ : *S* ×Σ → *S* ×Γ.

a) Give the line of code in Lex.cpp that establishes *s*0.

b) Where are the elements of *S* enumerated? Give the first three as they appear in the code.

c) What are the elements in the set Σ?

d) The code implements the state-transition function δ : *S* × Σ → *S* × Γ in two different ways depending on whether a token in Γ is to be emitted. Match the state-transition function to the code below by circling, labeling, and explaining how the code components provide the input *S*, Σ, the output *S*, and Γ for the *Comma* state and the input *S*, Σ, and the output *S*, for the *SawColon* state. Why is there no token emitted for the *SawColon* case in the switch statement?

State Lex::nextState() {

State result;

char character;

switch(state) {

case Start: result = getNextState(); break;

case Comma: emit(COMMA); result = getNextState(); break;

case Period: emit(PERIOD); result = getNextState(); break;

case SawColon:

character = input->getCurrentCharacter();

if(character == '-') {

result = Colon\_Dash;

input->advance();

} else { //Every other character

throw "ERROR:: in case SawColon:, Expecting '-' but found " + character + '.';

}

break;

case Colon\_Dash: emit(COLON\_DASH); result = getNextState(); break;

…