<https://github.com/VRPaul/FLCD-Lab4>

class Program  
{  
 */// <summary>  
 /// Runs the application by printing the list of available commands while*

*/// waiting for input only if the initialized automaton is valid  
 /// </summary>  
 /// <param name="args"></param>* static void Main(string[] args){}  
  
 */// <summary>  
 /// Gets the list of states from a given FA and prints it  
 /// </summary>  
 /// <param name="fa">A given FA</param>* private static void PrintStates(FiniteAutomaton fa){}  
  
 */// <summary>  
 /// Gets the alphabet from a given FA and prints it  
 /// </summary>  
 /// <param name="fa">A given FA</param>* private static void PrintAlphabet(FiniteAutomaton fa){}  
  
 */// <summary>  
 /// Gets the list of transitions from a given FA and prints it  
 /// </summary>  
 /// <param name="fa">A given FA</param>* private static void PrintTransition(FiniteAutomaton fa){}

*/// <summary>  
 /// Gets the list of final states from a given FA and prints it  
 /// </summary>  
 /// <param name="fa">A given FA</param>* private static void PrintFinalStates(FiniteAutomaton fa){}  
  
 */// <summary>  
 /// Checks if a sequence if valid. First we need to determine if the FA si a DFA.*

*/// If it's not we print a message  
 /// otherwise we start checking the sequence.  
 /// </summary>  
 /// <param name="fa"></param>* private static void CheckSequence(FiniteAutomaton fa){}  
}

*/// <summary>  
/// Representation of a finite automaton. Q: list of states, E: list of input symbols, q0: string of initial state  
/// F: list of final states, S: list of transitions  
/// </summary>*public class FiniteAutomaton  
{  
 public List<string> Q { get; }  
 public List<string> E { get; }  
 public List<string> F { get; }  
  
 public List<Tuple<Tuple<string,string>, string>> S { get; } = new();  
  
 private string q0;

*/// <summary>  
 /// Constructor gets a filePath, reads from it line by line and initializes*

*///Q,E,F,q0 and S  
 /// </summary>  
 /// <param name="faFilePath"></param>* public FiniteAutomaton(string faFilePath){}

*/// <summary>  
 /// Checks if the read FA is a valid one  
 /// </summary>  
 /// <returns>True if valid, false otherwise</returns>* public bool CheckIfValid(){}

*/// <summary>  
 /// Checks if a FA is DFA by making sure that there no more than one transitions*

*/// with a given input from a state  
 /// to another  
 /// </summary>  
 /// <returns>True if DFA, false otherwise</returns>* public bool CheckIfDFA(){}  
}

FA.in = "Q = ", state, { state }, "\n",

"E = ", symbol, { symbol }, "\n",

"q0 = ", state, "\n",

"F = ", state, { state }, "\n",

{" ", "(", state, ",", symbol, ") -> ", state};

state = "A" | ... | "Z";

symbol = "0" | ... | "9";