**CS 3350 – Automata**

**Summer 2021**

**Coding Assignment 1**

**Objective**

The objectives of this coding assignment are the following:

1. Learn how a Deterministic Finite Automaton (DFA) can be modeled and simulated using a modern programming language (in this case, Java).
2. Design a DFA that recognizes a given language and simulate its behavior on a set of input strings using the Java implementation.

**Context**

This assignment comes with two Java files:

* DFA.java: This file contains the logic necessary to model and simulate any DFA.
* Main.java: This file contains the project’s main method, which calls a set of methods that test the implementation of a DFA contained in DFA.java

Before solving the tasks presented in this assignment, read the code and familiarize yourself with the logic used to model and simulate a DFA. To facilitate this process, trace the following method: *runSampleDFATests.*

**Task 1**

Design a DFA that recognizes the following language: L = {w | w has at least one a and one b}, Σ = {a,b}

Examples of strings that should be accepted:

ba

ab

aaaab

bbba

aabba

Examples of strings that should not be accepted:

bbbb

aaaa

<empty string>

a

b

Once the DFA has been designed, model it and simulate it in Java by implementing the method *runAssignmentDFATests* (Main.java)*.* Use the code in *runSampleDFATests* as an example of how this can be done. Include at least 6 tests (3 input strings that are accepted by the DFA and 3 that are not accepted).

**Task 2**

Complete the implementation of the *isValid* method (DSF.java). This method should return true if and only if the 5-tuple that models the DSF is valid. This is what your method should do:

* If there is no start state (remember that the start state is stored in this.startState), return False
* If the start state is not one of the states contained in this.states, return False
* If any of the accept states is not one of the states contained in this.states, return False
* Every state in self.states should have a transition for every symbol in self.alphabet. If there are missing transitions in this.transitionFunction, return False
* If there are invalid transitions in this.transitionFunction, return False. A transition is invalid if: 1) the source state is not in this.states, and/or 2) the symbol is not in self.alphabet, and/or 3) the destination state is not in this.states.
* Return True if none of the previous conditions were met.

**Deliverables**

Make sure that your code is well documented by adding descriptive comments where necessary. Once you are ready, upload the two Java files using the submission page on Blackboard.