AERO 552: Aerospace Information Systems
Homework 2

Out Monday, September 18th, 2023

Due on CANVAS Thursday, September 28th, 2023, **11:59pm**

Instructions

Your solution should be submitted:

- a file `hw2.pdf` containing your solution to the handwritten problems;
- file `automaton.cpp` containing your solution to Exercise 1.

For the coding exercises, remember the following guidelines:

- **Do not use any C++ library unless explicitly authorized, except `iostream`, `fstream`, `cmath`, and `string`**
- High-level discussion of problems and directions with other students is fine, but exchanging solutions or code is not. If in doubt, please ask. **If you discussed with another student, please indicate so explicitly in your homework, with their name.**
- All code from this homework will be graded on CAEN Linux using the `g++` compiler – please test it on CAEN prior to submission.
- Include your code along with the examples you have tested your code on. Examples would typically be in a function `main`. Please **thoroughly test your code**: part of your grade will be based on how well you have tested your code, including corner cases. Moreover, many errors can be caught with good testing.
- Please submit code that compiles (on Linux) **without any warning or error**, and runs right away. We will take off points if your code does not run or if it produces warnings.
- Be careful with pointer and memory management, and do not forget to `delete` your memory, even in your test cases. Points will be taken off for memory leaks (forgetting to `delete`).
- Please do not reference or utilize any online code.

Exercise 1 (code) – Automaton (35 points)

Write a function `bool automaton(string file, string input)` that simulates the execution of a deterministic finite state automaton on an `input`, and returns `true` if and only if the automaton accepts `input`. Your should create a proper, clean data structure to store your automaton, and a helper function creating that data structure from reading the file.

An automaton will be stored in the file with name `file.txt`, following the self-explanatory format given by the following example. You can assume that only digits and letters will be used, but some states might require two characters. In a valid automaton, all possible transitions will be defined.

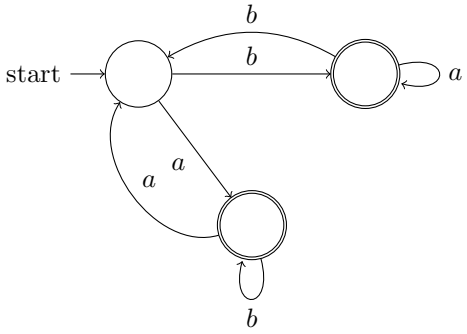
```
-- alphabet
a b
-- states
1 2 3
-- initial
1
-- final
1 3
-- transitions
1 -- a --> 2
1 -- b --> 1
2 -- a --> 3
2 -- b --> 2
3 -- a --> 3
3 -- b --> 3
```

Submit your code in a file `automaton.cpp`.

Exercise 2: DFA and regular expressions (45 points – 15 points each)

Construct a DFA that accepts each of the following languages. Specify your solutions both as a graphic with notational conventions used in the notes and lecture and by formally defining $\{Q, \Sigma, \delta, s, F\}$:

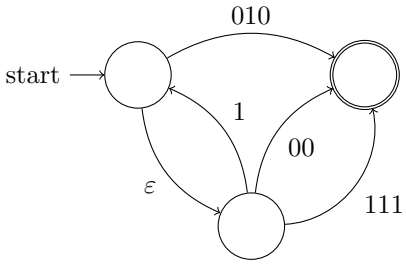
1. The language defined by the regular expression $(00 + 11)^*(01 + 10)(00 + 11)^*$ (18 points)
2. The language of the regular expression $(0(01)^*(1 + 00) + 1(10)^*(0 + 11))^*$ (18 points)
3. Give a regular expression equivalent to the following DFA: (20 points)



4. $L = \{x \mid x \in \{0, 1\}^*$ and any substring of length 4 contains at least two 1's$\}$.
   Note: this question is more difficult and requires some thinking. (21 points)

Exercise 3: NFA (15 points)

Convert the following NFA to a DFA. Show your conversion steps. What language L does this machine accept? (note that L must be specified in the form used in Exercise 1, question 4 below, or as a regular expression).



Bookkeeping (5 points)

In your `hw2.pdf`, indicate in a sentence or two how much time you spent on this homework, how difficult you found it subjectively, and what you found to be the hardest part. How deeply do you feel you understand the material it covers (0%–100%)? Any non-empty answer will receive full credit.