Problems

1. (4 points) Draw a DFA that accepts all strings over \{0, 1\} that contain an even number of 0s.

2. (4 points) Draw a DFA that accepts all strings over \{0, 1\} that contain an even number of 0s and an odd number of 1s.

3. (6 points) Draw a DFA that accepts all strings over \{0, 1\} that start with two 0s and end with two 1s.

4. (6 points) Draw a DFA that accepts the set of strings generated by the regular expression

   \(((00 + 11)^*11(00 + 11))^*)^*\).

5. (5 points) Give a regular expression that generates the set of strings from question 1.

6. (15 points) Write a computer program that simulates a deterministic finite automaton. Given the state table of a DFA and a string, your program should decide whether this string is accepted by the automaton. Use the programming language of your choice. A sample transcript defining and using 3 different automata (M, N, D) is below and continues on the back. (Your I/O need not emulate mine at all.)

   Submit both your program code and a sample transcript that uses at least these 3 automata. Be sure to include your name both in your program code and on your sample transcript. Upload both files using the CS200 submit script. You will need to use the submit script twice for the two files.

   If you search online for extra help, be sure you are searching for language features (e.g., how to create a list of tuples in Python) rather than code that answers the problem you are solving. Acknowledge (e.g., in comments) any sources that you do use. If you worked with someone else, including the TAs or the professor, please list their names as appropriate as well.

   CS200 HW11 Sample Solutions
   DFA transcript
   A Briggs

   >>> D = getDFA()
   State names: p q
   Alphabet: 0 1
   Enter transition table:
   0 1
   p: p q
   q: q p
   Start state: p
   Final states: q
   >>> simulate(D, ’1011’)
   True
   >>> simulate(D, ’10001’)
   False
   >>> simulate(D, ’01001001’)
   True
   >>>
>>> M = getDFA()
State names: p q r
Alphabet: 0 1
Enter transition table:

0  1
p: p q
q: r p
r: q r
Start state: p
Final states: p

>>> simulate(M, '1001')
True
>>> simulate(M, '1000')
False
>>> simulate(M, '10010')
True
>>> simulate(M, '11111')
False
>>> simulate(M, '1111')
True

>>> N = getDFA()
State names: {a,b,c,d,e,f,g}
Alphabet: {0,1}
Enter transition table:

0  1
a: b d
b: c e
c: b d
d: e f
e: e e
f: g d
g: f e
Start state: a
Final states: f

>>> simulate(N, '1100')
True
>>> simulate(N, '0101')
False
>>> simulate(N, '001100')
True
>>> simulate(N, '0000')
False
>>> simulate(N, '1111')
True

>>>