For assignment 1 I have developed a Finite State Automata simulator in Python 2.7. The first important step that took some time and thinking was the process of figuring out how to represent the automaton in the code. At first I thought that all the states and transitions look like a directed weighted graph and could be represented in this way but before proceeding with this idea I decided to first try to find a less complex solution and look into Python's built-in data structures. Upon closer study, I realized that it was possible to represent my FSA with just two classes and use a standard Python dictionary.

Thus, I have introduced a State class that has a designation (e.g. "q0"), a dictionary of possible transitions leading from this state (e.g. {'a', 'q1'} ) and two boolean values for the initial or final state. The value of the key of the dictionary is also an object of State class. The other class I introduced was the FSA class that has a number of methods for adding, getting (a list of all states, just one state with the parameter of its designation, initial state, final states) and printing the states of the automaton. It also has a method to check if the input string is accepted by this FSA. As a result it returns a string "True" or "False" and all the states it goes through while checking the input string. This is included in the fsa.py file.

One of the most complicated things was to parse the input file and deal with all the information that was contained there with all the newlines, spaces, curly brackets and parentheses and to construct an automaton from it. This is done in the program.py file along with reading and writing the input/output files.

To test the FSA I first used the provided test cases with minor changes, such as adding a space to the line, e.g. "ab baa", starting/ending the line with a space or adding elements that do not belong to the alphabet. Then I added another test case which we discussed in the lab - an automaton that gets a binary representation of a number and has to see if this number is divisible by three.

The input for this was the following:
{q0,q1,q2} {0,1} q0 {q0}
{q0(0)->q0,q0(1)->q1,q1(1)->q0,q1(0)->q2,q2(0)->q1,q2(1)->q2}
5
1010 1001 1111100111 10001 100110001001011001111111 which corresponds to the following decimal numbers: 10 9 999 17 9999999
The test showed that the FSA works even for large numbers as it returns 'True' for the last number.

To conclude, I really enjoyed doing the assignment. It was not too difficult but provided a lot of flexibility and allowed for greater understanding of the Finite State Automata and how they worked and could be implemented.