Link Github: https://github.com/stefan99x/FLCD/tree/master/Lab04

Transitions.py

```
class Transitions:
    def __init__(self):
        self.dictionary = {}

    def __str__(self):
        result = ""
        for start in self.dictionary.keys():
            for transition in self.dictionary[start]:
                aux = start + " --> " + transition[1] + " --> " +
transition[0] + "\n"
                result += aux

        return result

    def addTransition(self, start, end, value):
        '''
        :param start: starting position dictionary
        :param end: end position in dictionary
        :param value: as the name implies represents the value to be added
        :return: adds a new Transition to our transitions dictionary
        '''
        if start in self.dictionary.keys():
            self.dictionary[start].append((end, value))
        else:
            self.dictionary[start] = [(end, value)]

    def getTransitionsWithStart(self, start):
        '''
        :param start: position of the start
        :return: our dictionary from the given start
        '''
        if start not in self.dictionary.keys():
            return []
        return self.dictionary[start]
```

FiniteAutomatan.py

```
from Lab04.Transitions import Transitions


class FiniteAutomatan:
    def __init__(self, fileName):
        self.delta = Transitions()
        self.fileName = fileName
        self.Q = set()  # States
        self.E = set()  # Alphabet
        self.F = set()  # Transitions
        self.Q0 = None  # Initial State
        self.readFromFile()

    def readFromFile(self):
```

```
        '''
        as the name implies reads from a given file
        '''
        with open(self.fileName, 'r') as file:

            self.Q = file.readline().strip('\n').split(' ')
            self.E = file.readline().strip('\n').split(' ')
            self.F = file.readline().strip('\n').split(' ')
            self.Q0 = file.readline().strip('\n').split(' ')
            line = file.readline().strip('\n')

            while line:

                line = line.split(' ')
                for index in range(2, len(line)):
                    self.delta.addTransition(line[0], line[1], line[index])
                line = file.readline().strip('\n')

    def printStates(self):
        '''
        Print all the States
        '''
        print("States")
        for state in self.Q:
            print(state)

    def printAlphabet(self):
        '''
        Print all the Alphabet
        '''
        print("Alphabet of FA:")
        for value in self.E:
            print(value)

    def printInitialState(self):
        '''
        Print the initial State
        '''
        print(f"Initial state of FA: {self.Q0}")

    def printTransitions(self):
        '''
        Print the transitions
        '''
        print(f"Transitions: {self.delta}")

    def isDeterministic(self):
        '''
        Checks if it is deterministic
        :return: true or false
        '''
        for state in self.Q:
            transitions = self.delta.getTransitionsWithStart(state)
            for transition in transitions:
                uniqueTransitions = list(filter(lambda x: x[1] !=
transition[1], transitions))
                if len(uniqueTransitions) != len(transitions) - 1:
                    return False
        return True
```

```
    def isAccepted(self, string):
        '''
        Checks if it is accepted a given string
        :param string: input string
        :return: true or false
        '''
        if not self.isDeterministic():
            raise Exception("Not deterministic, sorry fam, quite sad init")
        currentState = self.Q0
        for index in range(len(string)):
            possibleTransitions =
self.delta.getTransitionsWithStart(currentState)
            currentValue = string[index]
            currentState = None
            for transition in possibleTransitions:
                if transition[1] == currentValue:
                    currentState = transition[0]
            if currentState is None:
                return False
        if currentState not in self.F:
            return False
        return True
```

Main.py

```
from Lab04.FiniteAutomata import *

def printMenu():
    '''
    Prints the Menu
    :return:
    '''
    print("{1}. Print States")
    print("{2}. Print Alphabet")
    print("{3}. Print Initial State")
    print("{4}. Print Transitions")

if __name__ == '__main__':
    finiteAutomata = FiniteAutomatan("fa.in")
    options = {1: finiteAutomata.printStates(), 2:
finiteAutomata.printAlphabet(),
               3: finiteAutomata.printInitialState(), 4:
finiteAutomata.printTransitions()}
    while True:
        printMenu()
        try:
            opt = int(input(">>"))
            if opt not in range(1, 5):
                pass
            options[opt]()
            print()
        except Exception as ex:
            print(ex)
            pass
```