Hoang Minh Le A18069640

1/ Alphabet: Σ = {0,1}
2/ Documentation for converting NFA to DFA: Both NFA and DFA are represented as follows:
List of states, alphabet, transition table, start state, and accept states.
In the transition table, there might be some [‘’], which does not stand for an empty string but stands for no input for that symbol.
Instruction for running the program: The program will ask the user to input the number of states and symbols in the alphabet. Simply input the number of states and symbols in the alphabet as the user wants. Then, the program will ask the user to input the names of the states in the NFA. Simply input the name for each state. Then it will ask the user to input the symbols for the alphabet. After that, the program will ask the user to input the start and accept states. Finally, for each state in the DFA, the user will have to input the transitions for each state, with each symbol. If there is no arrow for any symbol at any state, simply press enter without inputting anything.
***When adding transitions, if there is more than 1 state for 1 symbol, don’t put any blank after the comma. For example: for state q0 symbol 1, we can get to q0 and q1, simply input q0,q1, and enter. Don’t input q0, q1 or the program will not work properly. ***

Code:

```
#I used ChatGPT to help me with this task. Mainly I used it for
syntax.
#I asked: how do I print f-string in python
#ChatGPT answered: name = "John"
#age = 25
#print(f"Hello, my name is {name} and I am {age} years old.")

#I asked: what is [] and {} in python
#ChatGPT answered: Square Brackets []: These are used to define
lists, which are ordered collections of items.
#Lists can contain elements of different data types, and you can
access elements in a list by their index.
#Curly Braces {}: These are used to define dictionaries, which
are unordered collections of key-value pairs.
#In dictionaries, each key is associated with a value, and you
can use the key to look up the corresponding value.

#I asked: How do i convert from list to string
#ChatGPT answered: To convert a list to a string in Python, you
can use the `join()` method.
#my_list = ['a', 'b', 'c', 'd']
#my_string = ''.join(my_list)
#print(my_string)  # Output: 'abcd'
```

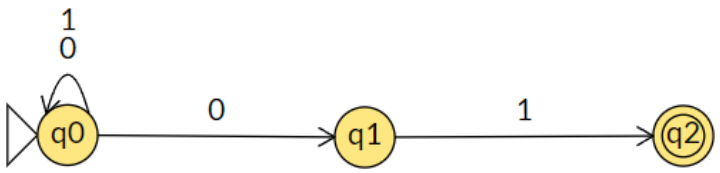
```
#NFA
statesNFA = []
num_states = int(input("Enter the number of states: "))
num_symbol = int(input("Enter the number of symbol in the
alphabet: "))
#states of the NFA
for state in range(num_states):
    while True:
        stateName = input("Enter state name (e.g., A, B, q1, q2):
")
        if stateName in statesNFA:
            print("State is already exists. Please enter a
different state.")
        else:
            statesNFA.append(stateName)
            break
sigma = []
# Input symbols in the alphabet
for symbol in range(num_symbol):
    while True:
        symbol = input("Enter your symbol for the alphabet: ")
        if symbol in sigma:
            print("Symbol is already exists. Please enter a
different symbol.")
        else:
            sigma.append(symbol)
            break
#Input accept states
acceptStatesInput = input("Enter accept states separated by comma
(e.g: 'q1,q2'): ")
acceptStatesNFA = acceptStatesInput.split(',')
#Input start state
startState = input("Enter start state: ")

#add transition for the NFA
transitions = {}
# Input transitions of the NFA
for state in statesNFA:
  for symbol in sigma:
    next_states_input = input(f"Enter next states from {state}
with symbol {symbol} separated by comma (e.g: 'q1,q2'): ")
```

```
    next_states = next_states_input.split(',')
    transitions[(state, symbol)] = next_states
#DFA
#adding transitions, states to the DFA
transitionsDFA = {}
statesDFA = [[startState]]  #The list to store all the states of
the DFA
for bigStates in statesDFA:
  for symbol in sigma:
    newStatesDFA = [] #The list to hold new states created during
the converting process
    for smallState in bigStates:
      if transitions[(smallState,symbol)] != ['']:  #because
input might have some '',stands for the arrow doesnt exist, not
empty string
        for s in transitions[(smallState,symbol)]:
          if s not in newStatesDFA:
            newStatesDFA += [s]
    transitionsDFA[(",".join(bigStates),symbol)] =
",".join(newStatesDFA)
    if newStatesDFA not in statesDFA: #if the new states are
already in the states list, we skip
      statesDFA.append(newStatesDFA)
#Find the accept states for the DFA
acceptStatesDFA = []
for bigStates in statesDFA: #iterate throught the each states in
the DFA
  for smallStates in bigStates: #iterate throught each states in
the big states that we just iterated
    if smallStates in acceptStatesNFA: #if the small state is the
accept state of the NFA, we will add the whole big state into the
accept state of DFA
      acceptStatesDFA.append(bigStates)
# Print the NFA information
print("NFA")
print("List of states: ",statesNFA)
print("Alphabet: ", sigma)
print("Transitions Table:")
for key, value in transitions.items():  #use a for loop to print
all the transitions of the DFA
  print(f"{key} -> {value}")
print("Start state: ",startState)
print("Accept states: ",acceptStatesNFA)
```

```
# Print the DFA information
print("DFA")
print("List of states: ",statesDFA)
print("Alphabet: ", sigma)
print("Transition Table:")
for key, value in transitionsDFA.items():
  print(f"{key} -> {{{value}}}")
print("Start state: ",startState)
print("Accept states: ",acceptStatesDFA)
```

3/ NFA Diagram
The alphabet of the NFA: {0,1}
The NFA has 3 states: q0, q1, and q2 where q2 is the accept state and q0 is the start state.



Screenshots of demonstration of running the program of the chosen NFA:

```
Enter the number of states: 3
Enter the number of symbol in the alphabet: 2
Enter state name (e.g., A, B, q1, q2): q0
Enter state name (e.g., A, B, q1, q2): q1
Enter state name (e.g., A, B, q1, q2): q2
Enter your symbol for the alphabet: 0
Enter your symbol for the alphabet: 1
Enter accept states separated by comma (e.g: 'q1,q2'): q2
Enter start state: q0
Enter next states from q0 with symbol 0 separated by comma (e.g: 'q1,q2'): q0,q1
Enter next states from q0 with symbol 1 separated by comma (e.g: 'q1,q2'): q0
Enter next states from q1 with symbol 0 separated by comma (e.g: 'q1,q2'):
Enter next states from q1 with symbol 1 separated by comma (e.g: 'q1,q2'): q2
Enter next states from q2 with symbol 0 separated by comma (e.g: 'q1,q2'):
Enter next states from q2 with symbol 1 separated by comma (e.g: 'q1,q2'):
```

```
NFA
List of states:  ['q0', 'q1', 'q2']
Alphabet:  ['0', '1']
Transitions Table:
('q0', '0') -> ['q0', 'q1']
('q0', '1') -> ['q0']
('q1', '0') -> ['']
('q1', '1') -> ['q2']
('q2', '0') -> ['']
('q2', '1') -> ['']
Start state:  q0
Accept states:  ['q2']


DFA
List of states:  [['q0'], ['q0', 'q1'], ['q0', 'q2']]
Alphabet:  ['0', '1']
Transition Table:
('q0', '0') -> {q0,q1}
('q0', '1') -> {q0}
('q0,q1', '0') -> {q0,q1}
('q0,q1', '1') -> {q0,q2}
('q0,q2', '0') -> {q0,q1}
('q0,q2', '1') -> {q0}
Start state:  q0
Accept states:  [['q0', 'q2']]
```