**예비프로젝트 2-1 -NFA to m-DFA 변환기**

20160389 원종하

1. 프로그램 제작 환경

OS: Windows 10 Home 64bit

언어: Python 3.6.1

Notepad++ v7.5.1로 작성.

2. 실행 방법

폴더 내의 eNFA2mDFA.py을 실행한다. 예시 입력들은 폴더 내의 example.txt에 있다.

3. 프로그램 설명

예비프로젝트 1-1에서 제작하였던 DFA를 변형하여 DFA와 NFA 모두를 적용 가능한 class FA를 만들고, 입력 형식대로 받은 입력을 토대로 FA type -NFA를 생성한다. 이 -NFA를 먼저 DFA로 변환한 후, 변환한 DFA를 다시 m-DFA로 변환하여 최종 결과인 m-DFA를 출력한다.

입력 방식 및 첫 화면은 Figure 1과 같다.

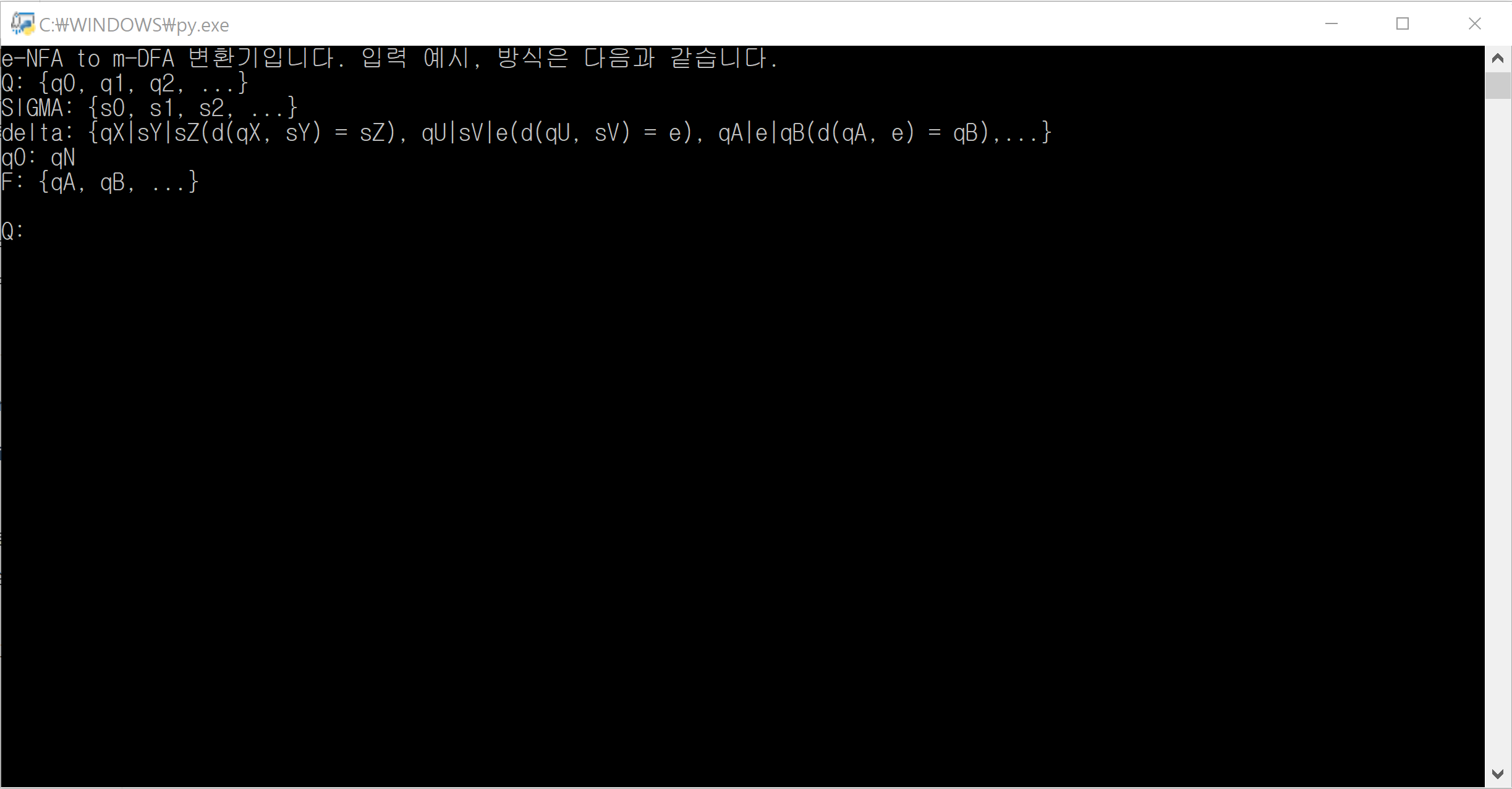


Figure 1

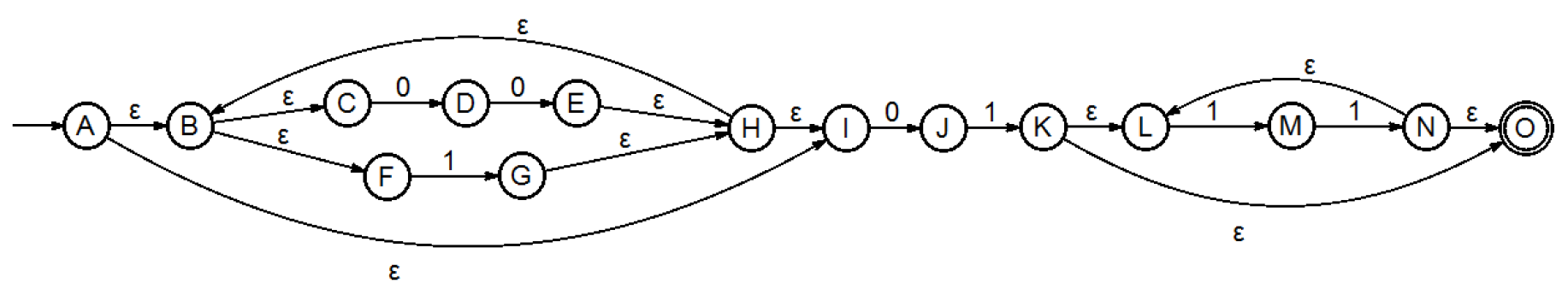


Figure 2

Figure 2와 같은 -NFA를 입력하려면, 차례대로 다음과 같이 입력하면 된다.

**{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O}**

**{0, 1}**

**{A|e|B, A|e|I, B|e|C, B|e|F, C|0|D, D|0|E, E|e|H, F|1|G, G|e|H, H|e|B, H|e|I, I|0|J, J|1|K, K|e|L, K|e|O, L|1|M, M|1|N, N|e|L, N|e|O}**

**A**

**{O}**

이 입력은 공백을 인식하므로 주의할 필요가 있다. 반점(,) 이후에는 항상 공백을 입력해야 한다.

예시로 든 입력의 출력 결과는 Figure 3과 같다.

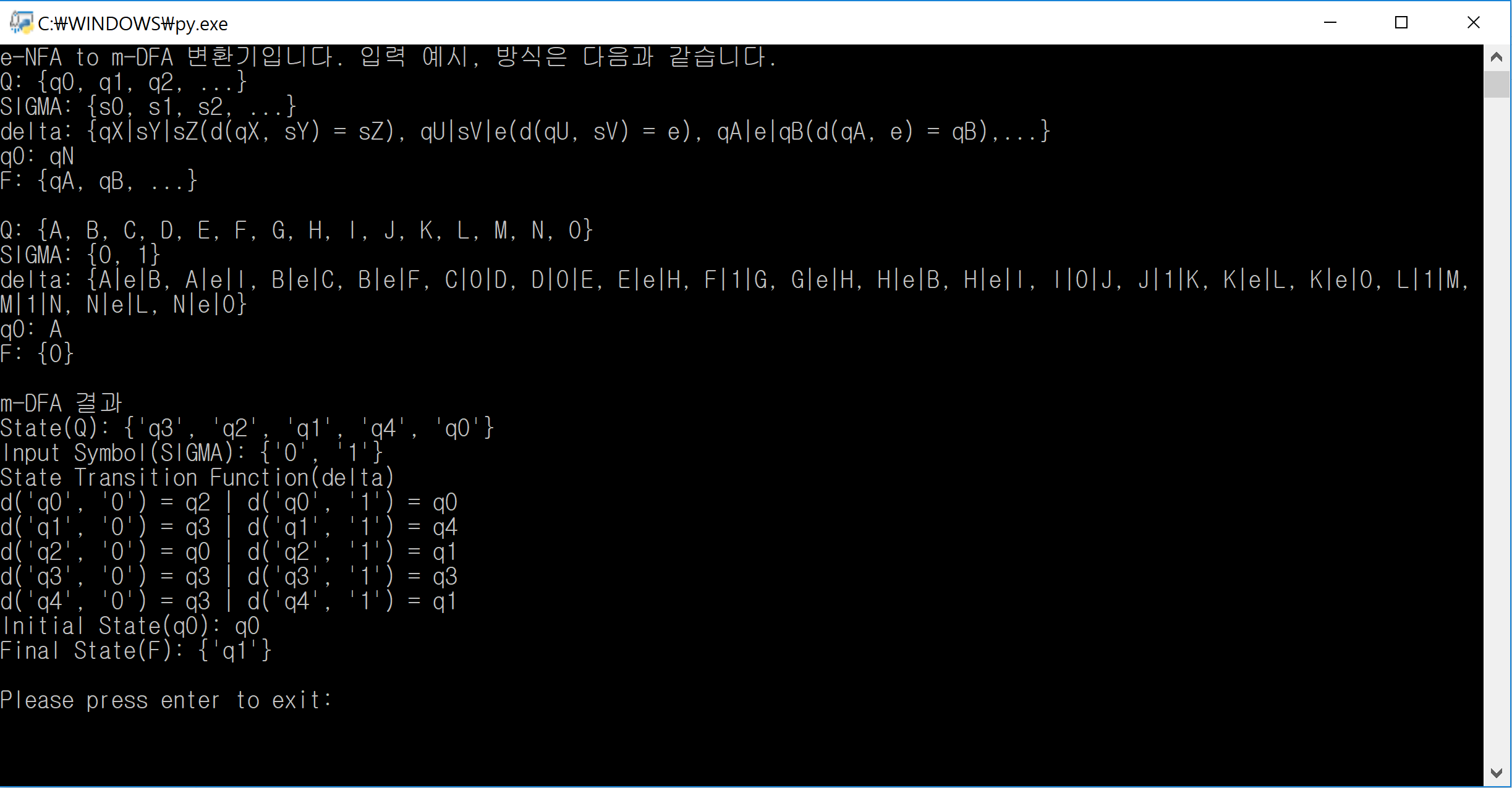


Figure 3

에서 은 실행 시마다 달라질 수 있지만, 모두 같은 DFA이다. Enter, 혹은 임의의 string을 입력한 후 enter하면 프로그램이 종료된다.

-NFA를 m-DFA로 변환하기 위해서는 한 단계 중간 과정을 거쳐야 한다. 우선 입력된 -NFA는 DFA로 변환한 후, 변환된 DFA를 다시 m-DFA로 변환하는 것이다. 우선 -NFA, DFA 변환은 Figure 4의 pseudo code를 참고하였다.

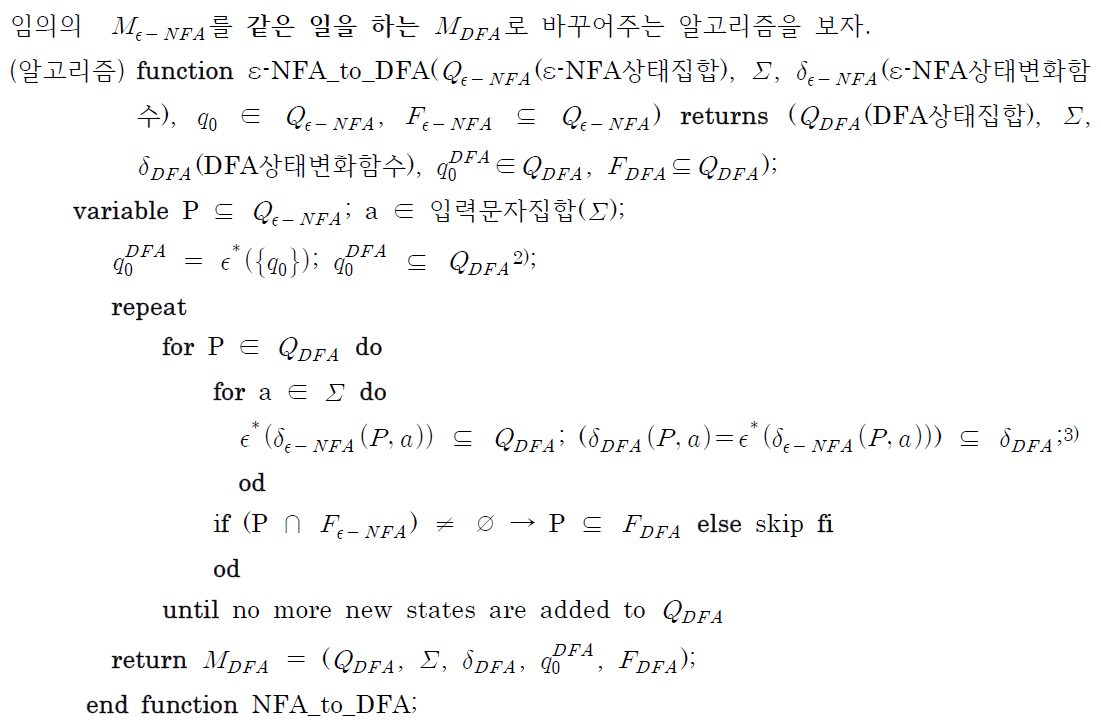


Figure 4

또한 DFA, m-DFA 변환은 Figure 5의 과정을 참조하였다.

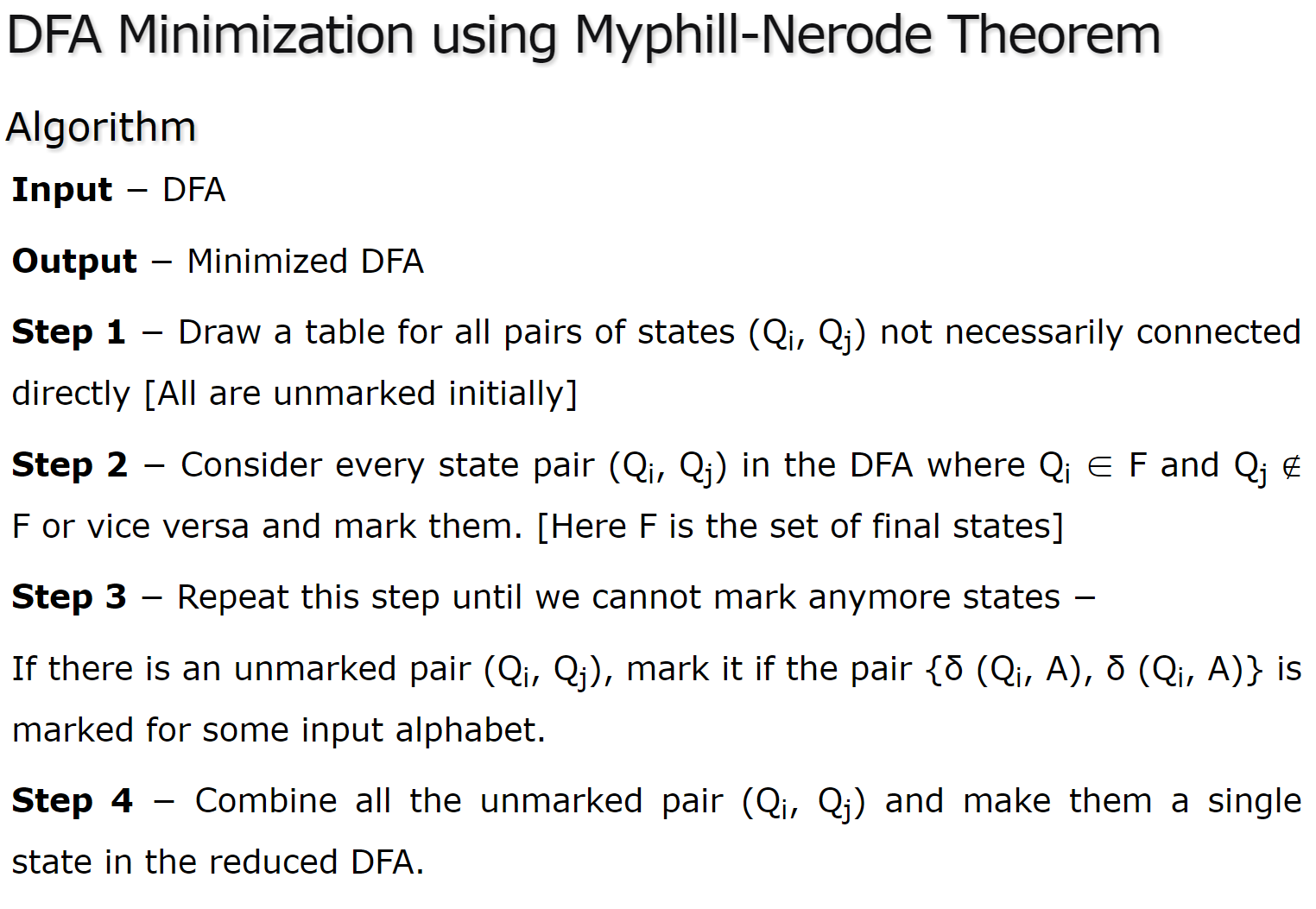


Figure 5

(출처는 <https://www.tutorialspoint.com/automata_theory/dfa_minimization.htm>이며, Step 3에 작성되어 있는 는 의 오기이다.)

4. 코드 설명

**FA**: 입력된 string을 DFA 혹은 NFA로 만드는 class

|  |  |
| --- | --- |
| 메서드 | 설명 |
| **\_\_init\_\_** | active: 실행시킬 수 있는 FA인가? (boolean)  run: 어떤 방식으로 입력을 받을 것인가? (boolean)  Qset(Q-set): 상태들의 유한 집합 (set)  Sset(Sigma-set): 입력 문자들의 유한 집합 (set)  ddic(delta-dictionary): 상태변화함수. (qN, sM) tuple을 key로, qM을 Value로 하는 dictionary (dict)  qzr(q-zero): 초기 상태 (str)  Fset(Final-set): 최종 상태들의 유한 집합 (set)    run이 1일 때 입력된 Q, SIG, delta, qz, Fin을 적당한 작업을 통해 Qset, Sset, ddic, qzr, Fset으로 변환한다.  1이 아닐 때는 모두 초기 상태로 설정한다. 이는 처음 선언에 데이터를 입력하지 않고, 나중에 직접적인 접근을 하기 위해서이다.  active는 정상적인 상황에서 True이며, ErrorMessage에 진입하였을 때 False가 된다.  Qset와 Sset는 중괄호 ({})을 슬라이싱한 후 ', '을 기준으로 스플릿하면 입력을 원소로 가지는 list가 되며, 그를 set으로 형변환 된다.  ddic은 마찬가지로 슬라이싱/스플릿 과정을 거친 후 in2df 메서드를 이용하여 dict type, 혹은 ErrorMessage(1 or 2)  qzr은 입력된 값이 Qset에 있다면 입력된 값을, Qset에 없다면 ErrorMessage(3)  Fset는 Fsetcheck 메서드를 통해 검사를 거친 후 set type, 혹은 ErrorMessage(4) |
| **in2df** | in2df: input을 dictionary type으로 변환하는 메서드  newdel은 {qX|sY|sZ(d(qX, sY) = sZ), qU|sV|e(d(qU, sV) = e), qA|e|qB(d(qA, e) = qB),...} 형식으로  입력된 input을 슬라이싱/스플릿하여 'X|Y|Z' 와 같은 형태의 string을 원소로 가지는 list이다  이 원소들을 다시 |을 매개로 스플릿하여 3개의 원소를 가지는 list로 만든 후, 몇 개의 검사를 거친다.  1. tmp[0](X)가 Qset에 없거나, tmp[2](Z)이 e가 아니면서 Qset에 없다면 ErrorMessage(1)  2. tmp[1](Y)이 Sset에 없거나 e가 아니라면 ErrorMessage(2)  이 검사를 거친 후 Error 조건을 만족하지 않는다면, (X, Y) tuple을 value로, d(X, Y)들이 원소인 set을 key로 dic에 넣고 return한다. |
| **Fsetcheck** | Fsetcheck: 입력(Fin)을 검사하여 정상적이라면 set을, 아니라면 ErrorMessage를 return  입력을 슬라이싱/스플릿 한 후, set type을 형변환하여  F - Qset가 공집합이라면 (F in Q) list를  공집합이 아니라면 (F not in Q) ErrorMessage(4)를 return한다 |
| **ErrorMessage** | ErrorMessage: num에 따라 알맞은 오류 메시지를 출력하는 메서드  active를 False로 바꿔준다. num은 다음과 같은 case를 의미한다.  1: not in in function  2: not in in function  3: not in  4: not in |

**set2str(set)**: set을 string으로 변환하는 함수. Set {‘a’, ‘b’, ‘c’}를 string “{a, b, c}”와 같이 변환한다.

**str2set(str)**: string을 set으로 변환하는 함수. String “{a, b, c}”를 set {‘a’, ‘b’, ‘c’}와 같이 변환한다.

**emove(eNFA)**: FA type eNFA의 state를 key로, state와 e(state)들을 원소로 하는 set을 value로 하는 dictionary를 return하는 함수.

**estar(emove)**: 함수 emove를 통해 생성된 dictionary emove를 이용하여 state를 key로, (state)를 value로 하는 dictionary를 return하는 함수.

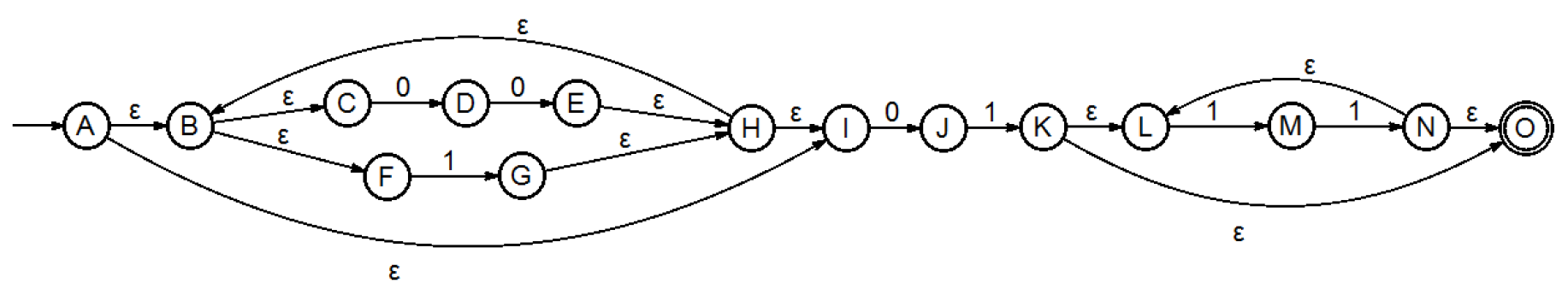
**eNFA2DFA(NFA)**: -NFA를 DFA로 변환하는 함수. 자세한 설명은 주석 참조.

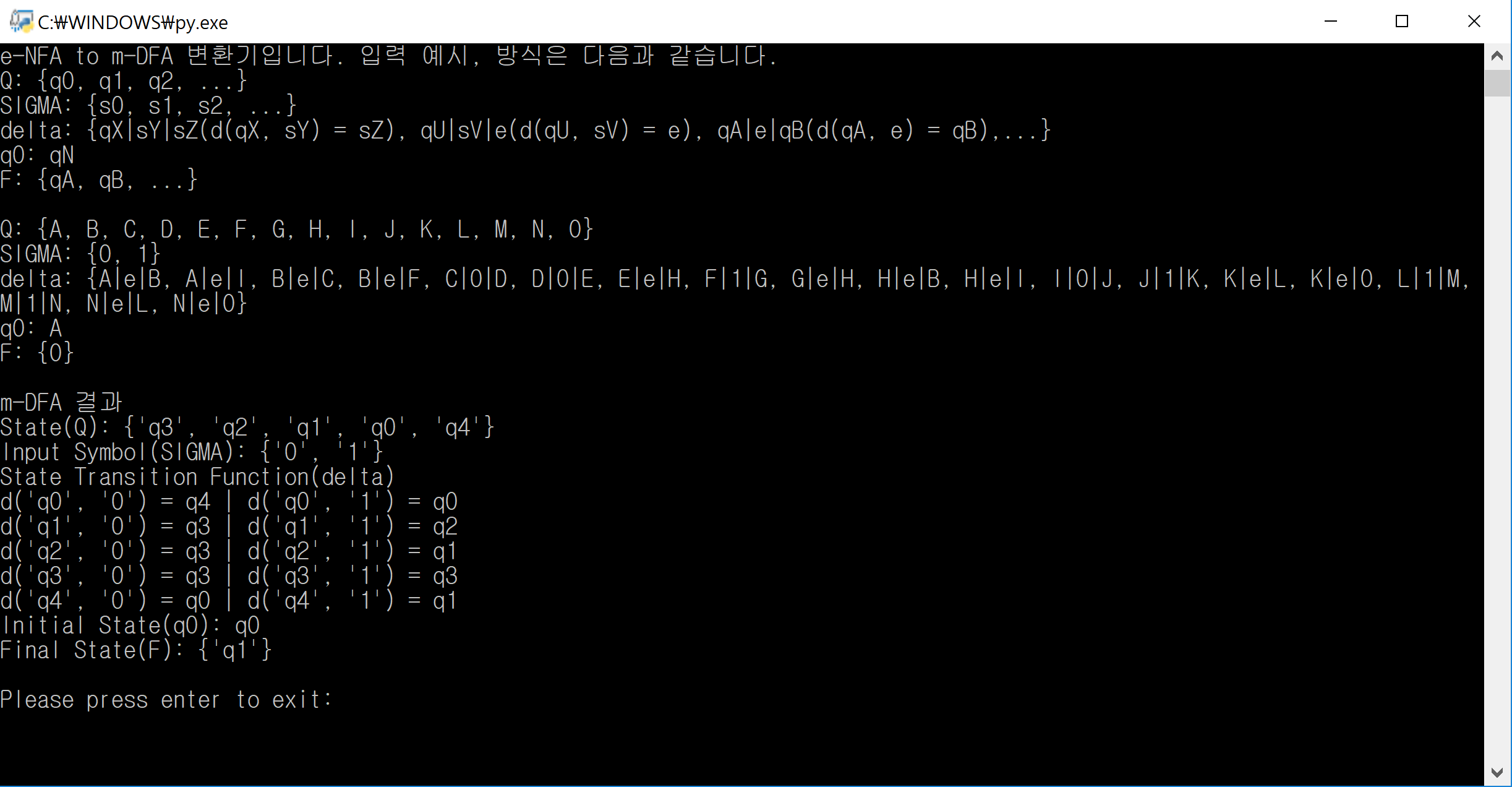
**DFA2mDFA(DFA)**: DFA를 m-DFA로 변환하는 함수. 자세한 설명은 주석 참조.

**printing(FA)**: parameter로 받은 FA를 state(), input symbol(), state transition function(), initial state(), final state() 순서대로 출력하는 함수.

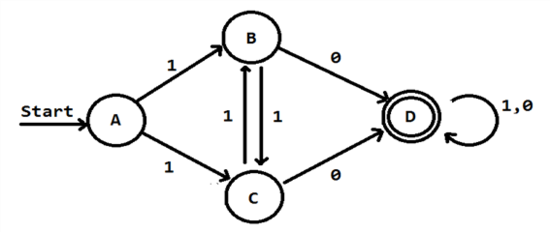
6. 입력 예시

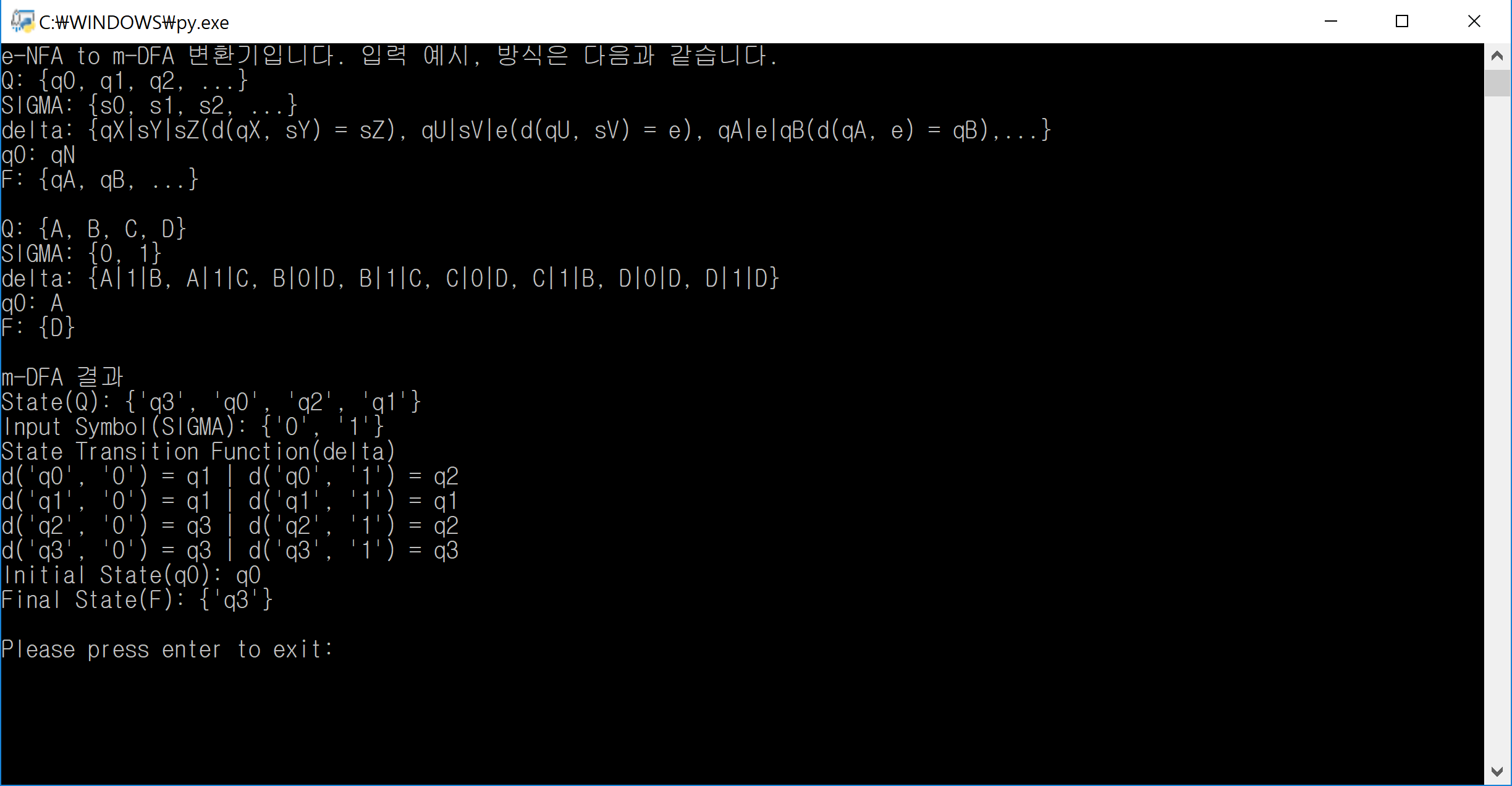
1) -NFA





2) NFA





3) DFA

