(예비프로젝트 1-1) DFA 시뮬레이터

제3주 화요일(9/13; 제4강 이후) 출제 15일 제5주 수요일(9/28) 까지 제출

상태변화함수 δ에서 부분함수를 허용하는 DFA $D = (Q, \Sigma, \delta, q_0, F)$를 **읽고**, 임의의 입력 문자열 $x \in \Sigma^*$에 대하여,

$x \in L(D)$ 이면 "네"를,

$x \notin L(D)$ 이면 "아니요"를

출력하는 DFA 시뮬레이터를 만든다.

이 예비 프로젝트는 본 프로젝트 1(한글모아쓰기 오토마타)과 연결될 부분프로젝트이다. 아주 쉽다.

하지만 상태변화함수 δ에서 부분함수를 허용하는 DFA $D = (Q, \Sigma, \delta, q_0, F)$를 입력하는 방법이 쉬워야, 앞으로 계속할 후속 프로젝트와의 **연결**이 쉬워질 것이고, 당신이 DFA를 사용하는 프로그램을 계속 만든다면 **재사용**도 쉬워 질 것이다.

제출 시 유의사항

작성할 프로그래밍 언어는 다음으로 제한한다. **C, C++, Python, Java.** 코드와 함께, 작성한 프로그램을 실행시키기 위한 **환경**, **컴파일 및 실행 방법**을 명시하여 README를 첨부해야 한다. 또한 작성한 프로그램은 아래에 명시된 방식으로 **DFA와 입력 문자열을 처리**해야 한다. 본 과제와 함께 첨부된 **입출력 테스트 케이스**를 참조하라.

입력 방식 및 입출력 테스트 케이스 설명

0. Readme

아래와 같이 DFA 입력 방식과 Input 문자열 입력 방식을 제한합니다. 다음 포맷을 지키지 않을 경우에 불이익이 있을 수 있습니다. 첨부한 ‘dfa.txt’, ‘input.txt’, ‘output.txt’ 파일은 프로젝트 수행에 있어서 자신의 코드를 테스트하기 위한 간단한 test case로 사용할 수 있습니다.

1. DFA 입력 방식

DFA를 입력하는 방식은 'dfa.txt' 파일을 읽어 결정합니다.

dfa.txt	
	// DFA가 $M = (Q, \Sigma, \delta, q_0, F)$로 표현될때
State	// Q는 **상태**(State)를 의미
q0,q1,q2,q3,q4	// 각 상태 q0~q4는 ,로 구분됨
Input symbol	// Σ는 **입력문자**(input symbol)를 의미
a,b	// 각 입력문자 a,b는 ,로 구분됨
State transition function	// δ는 **상태변화함수**를 의미
q0,b,q1	// 현 상태 (q0)에서 입력문자(b)를 보고 다음 상태(q1)을
q1,b,q2	의미하며 ,로 구분됨
q2,a,q3	
q3,b,q4	
q4,a,q4	
q4,b,q4	
Initial state	// $q_0 \in Q$는 **처음상태**를 의미
q0	
Final state	// $F \subseteq Q$는 **끝나는 상태**를 의미
q4	// 다수의 끝나는 상태가 존재하면 ,로 구분

2. 문자열 입력 방식

문자열을 입력하는 방식은 'input.txt' 파일을 읽어 결정합니다.

input.txt	
bb	// 입력받는 문자열은 **각 줄별로** DFA시뮬레이터의 결과를 출력
ababab	
bbabb	// DFA에 대한 결과는 '네'
aabaab	// DFA에 대한 결과는 '아니요'
	// 다음과 같은 **empty string**에 대한 결과도 출력 가능해야함
bababa	
bbabbaa	
bbabaab	
aaabbb	
aa	

3. Output 출력 방식

Input 문자열에 대한 출력 방식은 다음과 같이 'output.txt' 텍스트 파일 형식으로 출력하여야 합니다. 이때 입력 문자열과 일대일 대응되도록 합니다.

output.txt	
아니요	// 위 input 문자열에 대한 output 결과가 대응됨
아니요	
네	
아니요	
아니요	
아니요	
네	
네	
아니요	
아니요	