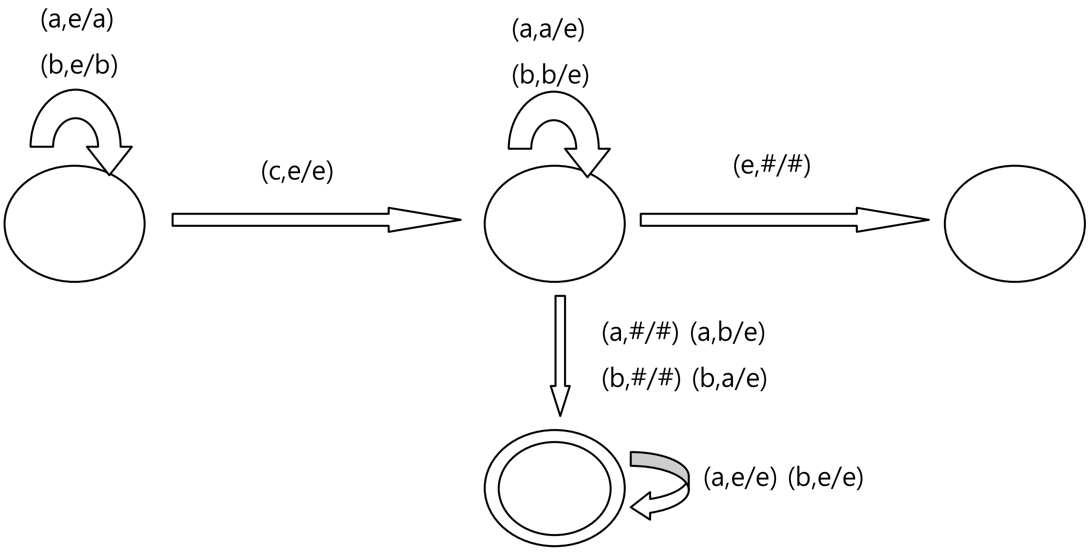
1.

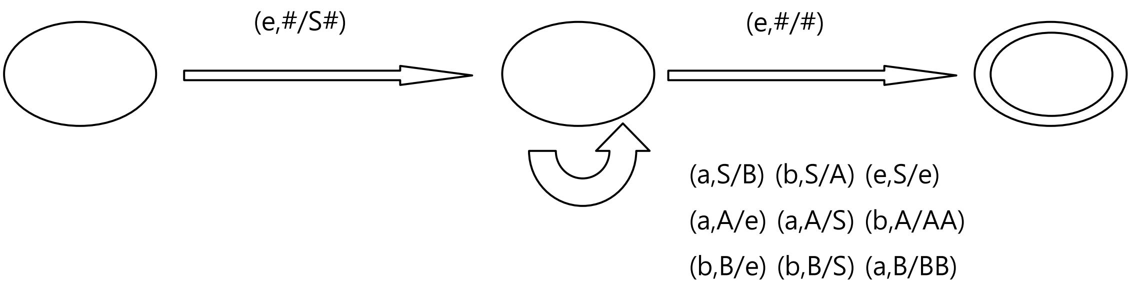
입력 문자 c를 기준으로 왼쪽에서는 PDA에 문자를 쌓고, 오른쪽에서는 그것을 검증하는 형태로 작업을 진행하면 된다. 단, $w_1 \neq w_2^R$은, "스택에 쌓아둔 문자열의 내용이 서로 다르다"만을 의미하는 것이 아니라, "좌우 문자열의 길이가 서로 다르다"의 경우 역시 포함함을 잊지 않아야겠다.



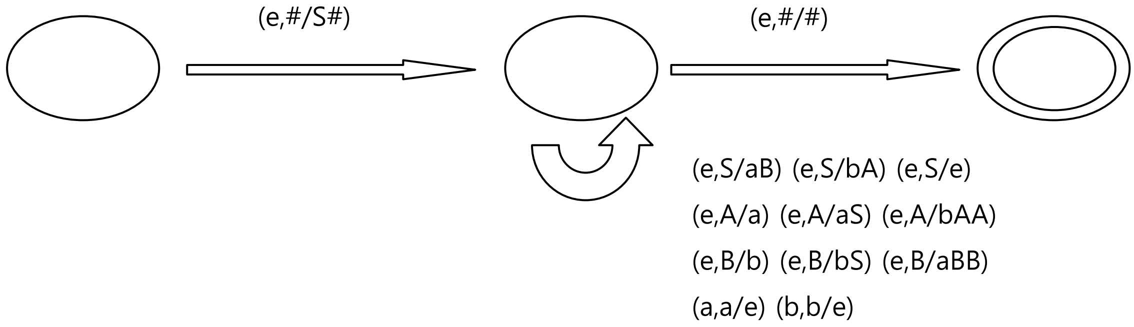
- PDA 10점, 설명 10점. PDA가 설명과 일치하지 않는 경우 설명만 틀린 것으로 간주.
- 잘못된 PDA 5점. 잘못된 PDA를 설명한 경우 설명도 5점.
- 잘못된 설명 5점. 논리적 비약은 3점 감점. 일부 오탈자 1점.

2.

Greibach 표준형으로 본다면 다음과 같은 PDA를 생성할 수 있다.



Greibach 표준형으로 보지 않고 PDA를 만들면 다음과 같은 PDA가 생성된다.



- PDA 10점, 설명 10점. PDA가 설명과 일치하지 않는 경우 설명만 틀린 것으로 간주.
- 잘못된 PDA 5점. 잘못된 PDA를 설명한 경우 설명도 5점.
- 잘못된 설명 5점. 논리적 비약은 3점 감점. 일부 오탈자 1점.

3.

편의상 2-tape TM을 가정한다. 문제에서 n이 주어졌을 때 n!을 계산하기 위해서는 다음과 같은 프로그램을 작성해야 한다.

```
tape1 ← n
tape2 ← 1
while (tape1 ≠ 0)
    tape2 ← tape2 ∗ tape1
    tape1 ← tape1 − 1
end while
output tape2
```

이 프로그램을 그대로 TM 형태로 옮기면, 다음과 같은 형태가 된다.

a. 입력으로 주어진 것을 tape1로 가정하고, 맨 처음 1에다 head를 놓는다. (tape1 ← n)
b. tape2는 #1#으로 초기화하고, 맨 처음 1에다 head를 놓는다. (tape2 ← 1)
c. tape1의 head가 #을 만나면, h로 이동한다. (while (tape1 ≠ 0)) 그 외의 경우, tape1의 head를 한 칸 오른쪽으로 옮긴다. 이때 tape1의 head가 #을 만나면 f로 이동한다.
d. [tape2의 가장 왼쪽 1을 b로 바꾸고, tape2 맨 마지막에 a를 붙이는 작업]을 tape2 가장 왼쪽 글자가 a가 될 때까지 반복한다. 그 후 tape1의 head를 한 칸 오른쪽으로 옮긴다. 이때 tape1의 head가 #을 만나면 f로 이동한다.
e. [tape2의 가장 왼쪽 a를 b로 바꾸고, tape2 맨 마지막에 1를 붙이는 작업]을 tape2 가장 왼쪽 글자가 1이 될 때까지 반복한다. 그 후 tape1의 head를 한 칸 오른쪽으로 옮긴다. 이때 tape1의 head가 #을 만나면 f로 이동한다.
f. tape2의 모든 b를 1로 바꾼다. (여기까지 tape2 ← tape2 ∗ tape1)
g. tape1의 맨 왼쪽 1을 0으로 바꾸고, 그 다음 글자에 head를 위치시킨 후 c로 돌아간다. (tape1 ← tape1 − 1)
h. tape2를 출력한다. (output tape2)

- 어떤 방법이든 정확하게 n!을 구하는 TM을 설명했다면 정답으로 인정
- 설명한 TM이 n!을 구하지 못하거나, 설명이 완전히 끝나지 않은 경우 10점.
- 논리적 전개를 흐트러뜨리는 논리적 비약 5점 감점. 일부 표기상 문제 1점 감점.

4.

tape를 두 개의 track으로 나누고, 각각의 칸에 alphabet을 입력할 수 있다고 가정하면 된다. 이 경우, 본래의 TM의 tape alphabet이 Σ에서 $(\Sigma \cup \bar{\Sigma}) \times (\Sigma \cup \bar{\Sigma})$가 된다고 생각하면 된다. 전이 역시 비슷한 방법으로 반영할 수 있다.

- 테이프를 접어서 처리하는 방법을 설명했다면 오답으로 간주.
- 두 방법 이외의 방법으로 2tape TM을 1tape으로 구현했다면 정답으로 인정.
- 오답, 미완성인 정답 10점 감점. 논리적 비약 5점 감점. 표기상 문제 1점 감점.

5.

$\overline{\mathrm{L_d}}$는 $\omega = \{0,1\}^*$ 중에서 TM의 부호가 아니거나, $\omega = \langle M \rangle$이지만 M이 ω에 대해 정지하지 않는 것들의 모임.

$\overline{\mathrm{L_d}}$가 재귀열거라면, 이를 정의하는 TM M, 즉 $\mathrm{L(M)} = \overline{\mathrm{L_d}}$인 M이 존재해야 한다. 그렇다면 <M>은 $\overline{\mathrm{L_d}}$에 속하는가, 속하지 않는가? $\langle M \rangle \in \overline{\mathrm{L_d}}$라면 $\langle M \rangle \in L(M)$이 되는데, 이는 $\overline{\mathrm{L_d}}$의 정의에 의해 $\langle M \rangle \notin \overline{\mathrm{L_d}}$를 의미하며, M의 정의에 모순된다. 그 반대의 경우도 마찬가지이다. 따라서 이는 M은 존재할 수 없으며, $\overline{\mathrm{L_d}}$는 재귀열거일 수 없다.

- $\overline{\mathrm{L_d}}$의 정의 10점, 증명 10점
- 틀린 정의 5점. “TM의 부호가 아닌 경우”를 빼먹은 경우는 틀린 것으로 간주.
- 미완성인 증명, 틀린 증명은 5점. 논리적 비약 3점 감점. 표기상 문제 1점 감점.

6.

$\mathrm{L_{CFL}}$ ={<M>:L(M) is a context free language}라 두겠다.

$\mathrm{L_{CFL}}$이 재귀열거라면, 이를 결정하는 TM $\mathrm{M_{CFL}}$이 있을 것이다. 한 CFL이 아닌 언어 L을 가정하고, $\mathrm{M_L}$을 이를 정의하는 TM이라 하겠다. (L=L($\mathrm{M_L}$)) 이제 ϵ-halting problem을 이용하겠다. $\mathrm{L_\epsilon}$의 입력 <M>이 주어지면, 이를 <M’>으로 변환한다. M’은,

- 주어진 입력 w를 따로 보관하고,
- ϵ에 대해 M을 돌리고,
- M이 정지하면 w에 대해 $\mathrm{M_L}$을 돌리는 TM이다.

이는 <M>과 <$\mathrm{M_L}$>에 다음 두 전이를 더해서 만들 수 있다.

- 입력 w를 따로 보관하고, 입력 테이프에서 지운 뒤, M의 초기 상태로 들어가는 전이
- M의 정지상태에서 시작해서 w를 입력 테이프로 옮기고 $\mathrm{M_L}$의 초기 상태로 들어가는 전이

그렇다면,

- M이 ϵ에 대해 정지한다면 M'은 M_L과 동일, 즉 $L(M') = L$
- M이 ϵ에 대해 정지하지 않는다면 $L(M') = \phi$일 것이다.

Φ은 CFL이고, L은 CFL이 아니므로, M'을 우리가 갖고 있는 M_{CFL}에 입력했을 때의 결과는,

- M_{CFL}이 L(M')을 CFL이라 답했다면, $L(M') = \phi$, 즉 M은 ϵ에 대해 정지하지 않는다
- M_{CFL}이 L(M')을 CFL이 아니라 답했다면, $L(M') = L$, 즉 M은 ϵ에 대해 정지한다

따라서 우리는 이를 이용하여 ϵ-halting problem을 해결할 수 있다. 이는 ϵ-halting problem이 계산불가라는 사실에 모순이다. 따라서 M_{CFL}은 존재할 수 없고, CFL은 계산불가이다.

- ϵ-halting problem 외의 다른 계산불가 문제를 이용하여 바르게 증명한 경우 정답 인정.
- Rice 정리에 의하여 계산불가라고 증명한 경우 잘못된 증명으로 간주.
- 잘못된 증명, 미완성인 증명 10점 감점. 논리적 비약 3점 감점, 표기상 문제 1점 감점.