

## 6주차 과제(연결리스트 실습)

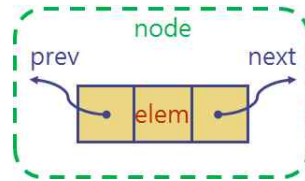
### ※ 입출력에 대한 안내

- 입출력 예시에서  $\mapsto$  이 후는 각 입력과 출력에 대한 설명이다.

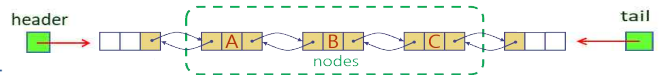
### 이중연결리스트 + 헤더 및 tail 노드 (문제 1 참고 내용)

#### 1. 연결리스트 구조

- 각 노드에 저장되는 정보
  - elem: 원소
  - prev: 이전 노드를 가리키는 링크
  - next: 다음 노드를 가리키는 링크



- 헤더 및 tail 노드
  - 데이터를 가지지 않는 더미(dummy) 노드



#### 2. 이중연결리스트의 초기화

- 초기에는 헤더 및 tail 노드만 존재
- $O(1)$  시간 소요

#### 3. 이중연결리스트의 순회

- 연결리스트의 모든 원소들을 방문(순회하면서 필요한 작업 수행. 예를 들면 출력)
- $O(n)$  시간 소요

#### 4. 이중연결리스트에서 삽입

- 이중연결리스트의 지정된 순위  $r$ 에 원소  $e$ 를 삽입
- $O(n)$  시간 소요

#### 5. 이중연결리스트에서 삭제

- 이중연결리스트로부터 지정된 순위  $r$ 의 노드를 삭제
- $O(n)$  시간 소요

[ 문제 1 ] 위에서 설명한 **이중연결리스트**를 이용하여 영문자 **리스트 ADT**를 구현하시오.

- 다음 네 가지 연산을 지원해야 함 (순위는 1부터 시작한다고 가정)
  - **add(r, e)** : **list**의 순위 **r**에 원소 **e**를 추가한다.
  - **delete(r)** : **list**의 순위 **r**에 위치한 원소를 삭제한다.
  - **get(r)** : **list**의 순위 **r**에 위치한 원소를 반환한다.
  - **print()** : **list**의 모든 원소를 저장 순위대로 공백없이 출력한다.
- ※ 순위 정보가 유효하지 않으면 화면에 에러 메시지 "invalid position"을 출력하고, 해당 연산을 무시한다.
- 입력에 대한 설명 (아래 입출력 예시 참조)
  - 각 연산의 내용이 한 줄에 한 개씩 입력되고, 한 개의 줄에는 연산의 종류, 순위, 원소 순서로 입력된다.
  - **연산의 종류**: 연산 이름의 맨 앞 영문자가 대문자 **A, D, G, P**로 주어진다.
  - **순위**: 양의 정수
  - **원소**: 영문자(대문자, 소문자 모두 가능)

입력 예시 1

출력 예시 1

5	↳ 연산의 개수: 5	
A 1 S	↳ add(1, 'S')	
A 2 t	↳ add(2, 't')	
A 3 r	↳ add(3, 'r')	
A 3 a	↳ add(3, 'a')	
P	↳ print()	Star      ↳ 연산 p에 의한 출력

입력 예시 2

출력 예시 2

9	↳ 연산의 개수: 9	
A 1 D	↳ add(1, 'D')	
A 2 a	↳ add(2, 'a')	
A 3 y	↳ add(3, 'y')	
D 1	↳ delete(1)	
P	↳ print()	ay
G 3	↳ get_entry(3)	invalid position
A 1 S	↳ add(1, 'S')	
P	↳ print()	Say
G 3	↳ get_entry(3)	y