탐색&시뮬레이션

(string, 1차원, 2차원 리스트 탐색)

- 회문 문자열 검사

- palindrome
- 대소문자를 구분하지 않는 경우 vs 대소문자를 구분하는 경우
  - 대소문자를 구분하지 않는 경우 대문자화(upper()) 또는 소문자화(lower())를 통해 문자열을 모두 같은 형태로 만든다.
- s: 문자열 -> **s[::-1]**은 문자열을 **reverse한 형태로 출력**함

- 숫자만 추출

- **s: 문자열 -> for x in s: 문**을 통해 문자열 s의 **문자 하나하나에 접근이 가능**해진다.

- 카드 역배치

- 두 변수의 값을 교환하는 방법
  - **a, b = b, a**
- 숫자를 저장하는 리스트 초기화 방법: a = list(range(n))

- 두 리스트 합치기

- sort() 함수 이용
  - 일반적으로 **시간 복잡도가 nlogn**
  - but, **이미 정렬된 list**의 경우 **시간복잡도가 n**
- 두 리스트의 index를 가리키는 pointer 변수 p1, p2를 선언
  - 두 리스트의 시작부터 원소들을 비교해가면서 작은 원소를 새로운 리스트에 삽입!
  - 두 리스트 중 포인터 변수가 하나의 리스트의 끝에 도착하면 다른 리스트의 나머지 모든 원소를 삽입!

- **수들의 합**
  - ■ 연속적인 부분 수열
  - ■ lt = 0, rt = 1로 주어진 리스트의 포인터 변수를 선언
- **격자판 최대합**
- **사과나무(다이아몬드)**
  - ■ s(시작), e(끝) 변수로 범위 설정
  - ■ **다이아몬드 형태**는 n // 2(행의 중간)에서 시작
- **곶감(모래시계)**
  - ■ s(시작), e(끝) 변수로 범위 설정
  - ■ **모래시계 형태**는 s = 0, e = n – 1 (행의 처음과 끝에서 시작)
- **봉우리**
  - ■ **all 함수**의 특징 기억
    - ◆ 함수 내의 **인자가 참이면 True를 return 거짓이면 False를 return**
  - ■ **dx = [-1, 0, 1, 0], dy = [0, 1, 0, -1]** -> 어떤 좌표에 대해 상하좌우 좌표를 구할 수 있음
- **스토쿠 검사**
  - ■ 행, 열, 3x3 격자판에 대해 각각 일차원 리스트 check를 선언
  - ■ 1~9까지 숫자를 확인하면 check 리스트에서 해당 index에 1을 저장.
  - ■ 중복되는 경우 한 곳이 0으로 남아있게됨. → check 리스트의 합이 9가 아니면 중복이 존재한다는 것을 알 수 있음!
  - ■ 3x3 격자판 탐색 → 9개 그룹을 탐색하는 for문 2개 + 각 그룹 내에서 9칸을 탐색하는 for문 2개 = 4중 for문 사용

- **격자판 회문수**

  - python의 slice 기능 기억하기 → x는 리스트나 문자열 - **x[i:i+5] = x의 index i부터 i+4까지를 가리킴**
  - 리스트나 문자열 x에 대하여 **x[::-1]**은 현재 x의 역순을 나타냄!