끝말잇기

`unordered_set`

- C++의 표준 라이브러리
- 고유한 원소들을 저장하는 컨테이너
- 중복을 허용하지 않으며, 각 원소는 유일해야함
- 해시 테이블을 기반으로 하여 원소를 저장

→ 원소의 삽입, 삭제, 검색 등의 연산이 평균적으로 O(1)의 시간 복잡도를 가

주요 특징

- **중복 허용하지 않음**: 동일한 원소를 두 번 이상 저장할 수 없음
- **빠른 검색**: 해시 테이블을 사용하여 평균적으로 매우 빠른 검색 속도를 제공
- **순서 없음**: 원소들이 저장되는 순서에 대한 보장이 없

주요 메서드

- `insert()`: 원소를 추가
- `erase()`: 원소를 제거
- `find()`: 원소를 검색
- `size()`: 저장된 원소의 개수를 반환

```
문제 설명
1부터 n까지 번호가 붙어있는 n명의 사람이 영어 끝말잇기를 하고 있습니다. 영

1번부터 번호 순서대로 한 사람씩 차례대로 단어를 말합니다.
마지막 사람이 단어를 말한 다음에는 다시 1번부터 시작합니다.
앞사람이 말한 단어의 마지막 문자로 시작하는 단어를 말해야 합니다.
이전에 등장했던 단어는 사용할 수 없습니다.
한 글자인 단어는 인정되지 않습니다.
다음은 3명이 끝말잇기를 하는 상황을 나타냅니다.
```

```
tank → kick → know → wheel → land → dream → mother → robot →

위 끝말잇기는 다음과 같이 진행됩니다.

1번 사람이 자신의 첫 번째 차례에 tank를 말합니다.
2번 사람이 자신의 첫 번째 차례에 kick을 말합니다.
3번 사람이 자신의 첫 번째 차례에 know를 말합니다.
1번 사람이 자신의 두 번째 차례에 wheel을 말합니다.
(계속 진행)
끝말잇기를 계속 진행해 나가다 보면, 3번 사람이 자신의 세 번째 차례에 말한 

사람의 수 n과 사람들이 순서대로 말한 단어 words 가 매개변수로 주어질 때,

제한 사항
끝말잇기에 참여하는 사람의 수 n은 2 이상 10 이하의 자연수입니다.
words는 끝말잇기에 사용한 단어들이 순서대로 들어있는 배열이며, 길이는 n 이
단어의 길이는 2 이상 50 이하입니다.
모든 단어는 알파벳 소문자로만 이루어져 있습니다.
끝말잇기에 사용되는 단어의 뜻(의미)은 신경 쓰지 않으셔도 됩니다.
정답은 [ 번호, 차례 ] 형태로 return 해주세요.
만약 주어진 단어들로 탈락자가 생기지 않는다면, [0, 0]을 return 해주세요.
입출력 예
n   words   result
3   ["tank", "kick", "know", "wheel", "land", "dream", "mothe
5   ["hello", "observe", "effect", "take", "either", "recogni
2   ["hello", "one", "even", "never", "now", "world", "draw"]
```

```
#include <string>
#include <vector>
#include <iostream>
#include <unordered_set>
using namespace std;

vector<int> solution(int n, vector<string> words) {
     vector<int> answer(2, 0);  // 초기 값으로 0을 가지는 벡터를 생
```

```
    unordered_set<string> wordSet;  // 중복된 단어를 체크하기 위한
    char prior = words[0][0];  // 첫 단어의 첫 글자로 초기화

    for (int i = 0; i < words.size(); i++) {
        int player = (i % n) + 1;  // 현재 플레이어 번호
        int round = (i / n) + 1;   // 현재 라운드

        // 이전 단어의 마지막 문자와 현재 단어의 첫 문자 비교
        if (i > 0 && words[i][0] != prior) {
            answer[0] = player;
            answer[1] = round;
            break;
        }

        // 현재 단어가 이미 사용된 단어인지 체크
        if (wordSet.find(words[i]) != wordSet.end()) {
            answer[0] = player;
            answer[1] = round;
            break;
        }

        // 현재 단어를 집합에 추가하고 prior를 현재 단어의 마지막 문자로
        wordSet.insert(words[i]);
        prior = words[i].back();
    }

    return answer;
}
```