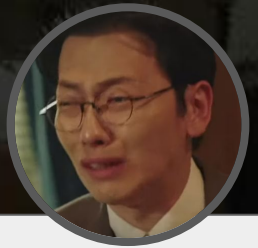
2019 KAKAO BLIND RECRUITMENT
오픈채팅방

Written by Young-in Byeon
GitHub : https://github.com/qus0in



응용
(확장성)

최적화
(효율성)

구현
(정확성)

등장인물

문석구

대마유통에 인턴으로
입사한 사원. 하지만
출근 첫날부터 알고리즘
문제에 봉착하는데…

조미란

문석구의 사수로,
마케팅은 물론
프로그래밍에도 능한
똑.부 타입의 대리

정복동

천리마마트의 사장,
부임한지 얼마 안 되어
직원 관리에 어려움을
겪고 있는 중이다.

빠야족

천리마마트에서 일하는
외국인 노동자.
트렌드에 민감하여
자주 별명을 바꾼다.

Chapter 1

아닌 밤중에 오픈채팅방

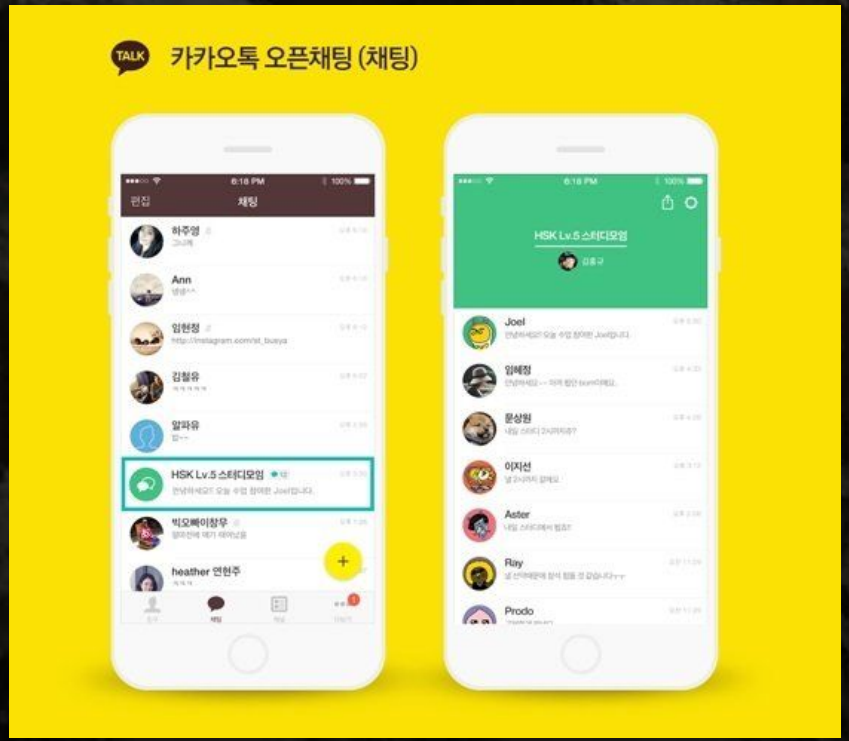
안녕하십니까! 이번에 인턴으로 입사하게 된 문.석.구라고 합니다. 잘 부탁드립니다!

안녕하세요, 석구씨. 같이 일하게 된 조미란 대리라고 합니다. 반가워요.

제가 듣기로는 프로그래밍 경력이 있다고
들었는데... 이 문제 한 번 풀어볼래요?

넵, 알겠습니다.
(괜히 자소서에 적었나? ㅠㅠ)

프로그래머스


카카오톡 오픈채팅 (채팅)
편집
채팅
하주영
Ann
임현정
김철유
알파유
HSK Lv.5 스터디모임
빅오빠이창우
heather 연현주
HSK Lv.5 스터디모임
Joel
임혜정
문상원
이지선
Aster
Ray
Prodo

문제 분석

[분석]

- 기능은 세 가지 -> Enter, Leave, Change
- " "으로 split해서 기능부분, id부분, 닉네임 부분으로 나눈다
- enter, change를 바탕으로 id(key)와 닉네임(value)을 HashMap으로 저장한다
- enter, leave + id 조합으로 message를 구성한다
- id를 닉네임으로 바꾸고, 기능별 멘트를 반영한 최종 message를 return한다

실제 코드

```
class Solution {
    public String[] solution(String[] record) {
        String[] div = null; // whitespace 기준으로 나눈 배열을 받아줄 자리
        HashMap<String, String> map = new HashMap<>(); // id (key) : 닉네임 (value)를 받아줄 HashMap
        ArrayList<String[]> list = new ArrayList<>(); // 디스플레이 될 기능 목록들 (배열 변환 전)
        String msg = null; // 디스플레이 될 메시지를 받아주는 자리
        for (String r : record) {
            div = r.split(" "); // " "로 나눠줌
            if (div[0].equals("Change")) { // Change 기능일 경우
                map.put(div[1], div[2]); // 이전에 id에 저장되었던 닉네임을 바꿔준다
            } else { // Enter, Leave 기능일 경우
                if (div[0].equals("Enter")) {
                    map.put(div[1], div[2]); // 새롭게 key 생성
                    msg = "님이 들어왔습니다."; // Enter 시 message
                } else {
                    msg = "님이 나갔습니다."; // Leave 시 message
                }
                // id와 메시지를 리스트에 추가
                list.add(Arrays.asList(div[1], msg).toArray(new String[2]));
            }
        }
        // stream-map을 통해 마지막으로 id와 닉네임을 일괄 변환 후 배열로 리턴
        return list.stream().map(v -> map.get(v[0]) + v[1]).toArray(String[]::new);
    }
}
```

테스트 결과

테스트 1

입력값 〉 ["Enter uid1234 Muzi", "Enter uid4567 Prodo", "Leave uid1234", "Enter uid1234 Prodo", "Change uid4567 Ryan"]

기댓값 〉 ["Prodo님이 들어왔습니다.", "Ryan님이 들어왔습니다.", "Prodo님이 나갔습니다.", "Prodo님이 들어왔습니다."]

실행 결과 〉 테스트를 통과하였습니다.

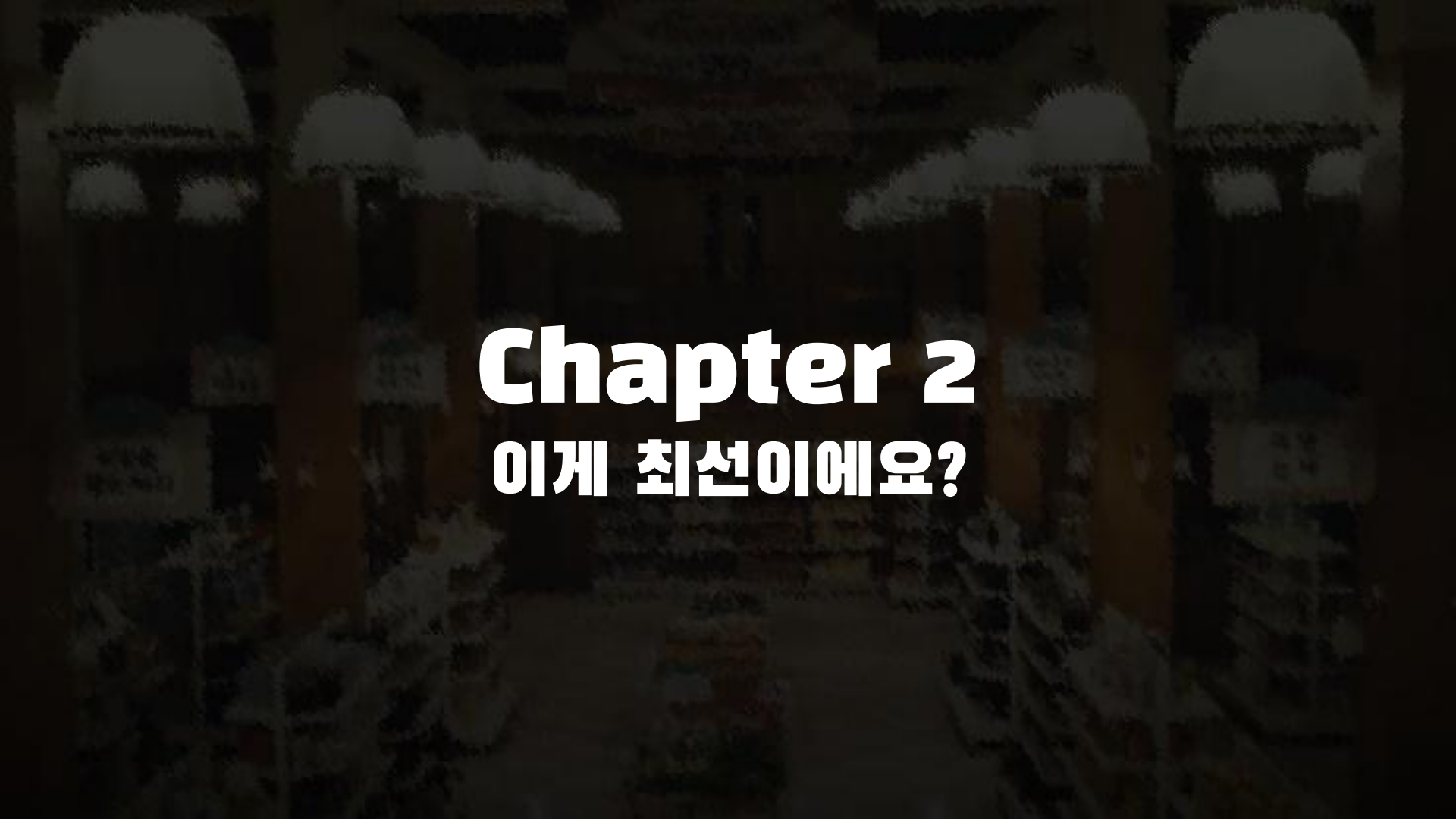
테스트 결과 (~˘▾˘)~

1개 중 1개 성공

샘플 테스트 케이스를 통과했다는 의미로, 작성한 코드가 문제의 정답은 아닐 수 있습니다.
(샘플 테스트 케이스는 [테스트 케이스 추가하기] 버튼을 통해 확인하실 수 있습니다.)

테스트 결과

테스트 1	통과 (38.57ms, 56.1MB)	테스트 17	통과 (38.30ms, 57.5MB)
테스트 2	통과 (36.41ms, 56.2MB)	테스트 18	통과 (35.69ms, 54.2MB)
테스트 3	통과 (35.90ms, 55.4MB)	테스트 19	통과 (61.89ms, 57.7MB)
테스트 4	통과 (33.47ms, 55.6MB)	테스트 20	통과 (45.23ms, 58.3MB)
테스트 5	통과 (49.77ms, 57.9MB)	테스트 21	통과 (47.07ms, 57.3MB)
테스트 6	통과 (64.73ms, 57.8MB)	테스트 22	통과 (48.31ms, 57.2MB)
테스트 7	통과 (47.56ms, 57.5MB)	테스트 23	통과 (51.49ms, 55.9MB)
테스트 8	통과 (57.73ms, 55.6MB)	테스트 24	통과 (49.93ms, 58.3MB)
테스트 9	통과 (78.20ms, 57.8MB)	테스트 25	통과 (281.92ms, 153MB)
테스트 10	통과 (47.47ms, 56.4MB)	테스트 26	통과 (285.11ms, 159MB)
테스트 11	통과 (40.29ms, 58.1MB)	테스트 27	통과 (356.71ms, 158MB)
테스트 12	통과 (43.95ms, 56.3MB)	테스트 28	통과 (333.32ms, 158MB)
테스트 13	통과 (52.83ms, 56.6MB)	테스트 29	통과 (314.67ms, 157MB)
테스트 14	통과 (74.16ms, 57.6MB)	테스트 30	통과 (323.22ms, 156MB)
테스트 15	통과 (35.82ms, 55.9MB)	테스트 31	통과 (297.93ms, 162MB)
테스트 16	통과 (35.72ms, 56MB)	테스트 32	통과 (290.20ms, 138MB)


Chapter 2
이게 최선이에요?

조미란 대리님, 코딩 완료했습니다.
검토 부탁드립니다.

수고했어요. 석구씨.
그런데… 호율성 테스트는 하셨나요?

구현만 되었다고 끝이 아니에요.
유저들은 1초 기다리는 것도 싫어한답니다.

네...엥
(뭔가... 잘못 걸린 것 같다.)

개선점 분석

[개선]

- StringTokenizer, StringBuilder 등 특화 클래스 사용
- stream을 쓰는 대신, for each문을 사용
- 각 for문 당 if문을 최소화 및 삼항연산자 사용

- if-not을 if-break-else 구조로 바꾸는 등의
  추가적인 시도를 하였으나 유의미한 차이를 발견하지 못했음

개선 코드

```java
class Solution {
    public String[] solution(String[] record) {
        StringTokenizer tokenizer = null;
        String cmd = null;
        HashMap<String, String> idMap = new HashMap<>();
        for (String r : record) {
            tokenizer = new StringTokenizer(r, " ");
            cmd = tokenizer.nextToken();
            if(!cmd.equals("Leave")) {
                idMap.put(tokenizer.nextToken(), tokenizer.nextToken());
            }
        }
        ArrayList<String> answer = new ArrayList<>();
        for (String r : record) {
            tokenizer = new StringTokenizer(r, " ");
            cmd = tokenizer.nextToken();
            if(!cmd.equals("Change")) {
                answer.add(new StringBuilder()
                                .append(idMap.get(tokenizer.nextToken()))
                                .append("Enter".equals(cmd) ? "님이 들어왔습니다." : "님이 나갔습니다.")
                                .toString());
            }
        }
        return answer.toArray(new String[answer.size()]);
    }
}
```

결과 비교

테스트 1	통과 (38.57ms, 56.1MB)	테스트 17	통과 (38.30ms, 57.5MB)
테스트 2	통과 (36.41ms, 56.2MB)	테스트 18	통과 (35.69ms, 54.2MB)
테스트 3	통과 (35.90ms, 55.4MB)	테스트 19	통과 (61.89ms, 57.7MB)
테스트 4	통과 (33.47ms, 55.6MB)	테스트 20	통과 (45.23ms, 58.3MB)
테스트 5	통과 (49.77ms, 57.9MB)	테스트 21	통과 (47.07ms, 57.3MB)
테스트 6	통과 (64.73ms, 57.8MB)	테스트 22	통과 (48.31ms, 57.2MB)
테스트 7	통과 (47.56ms, 57.5MB)	테스트 23	통과 (51.49ms, 55.9MB)
테스트 8	통과 (57.73ms, 55.6MB)	테스트 24	통과 (49.93ms, 58.3MB)
테스트 9	통과 (78.20ms, 57.8MB)	테스트 25	통과 (281.92ms, 153MB)
테스트 10	통과 (47.47ms, 56.4MB)	테스트 26	통과 (285.11ms, 159MB)
테스트 11	통과 (40.29ms, 58.1MB)	테스트 27	통과 (356.71ms, 158MB)
테스트 12	통과 (43.95ms, 56.3MB)	테스트 28	통과 (333.32ms, 158MB)
테스트 13	통과 (52.83ms, 56.6MB)	테스트 29	통과 (314.67ms, 157MB)
테스트 14	통과 (74.16ms, 57.6MB)	테스트 30	통과 (323.22ms, 156MB)
테스트 15	통과 (35.82ms, 55.9MB)	테스트 31	통과 (297.93ms, 162MB)
테스트 16	통과 (35.72ms, 56MB)	테스트 32	통과 (290.20ms, 138MB)

테스트 1	통과 (1.95ms, 52.5MB)	테스트 17	통과 (3.09ms, 52.2MB)
테스트 2	통과 (1.93ms, 52.4MB)	테스트 18	통과 (2.88ms, 52.3MB)
테스트 3	통과 (2.33ms, 50.1MB)	테스트 19	통과 (12.09ms, 55MB)
테스트 4	통과 (2.51ms, 51.1MB)	테스트 20	통과 (16.81ms, 52.9MB)
테스트 5	통과 (18.82ms, 54.9MB)	테스트 21	통과 (10.05ms, 51.1MB)
테스트 6	통과 (10.02ms, 53.7MB)	테스트 22	통과 (10.19ms, 51.2MB)
테스트 7	통과 (10.73ms, 52.9MB)	테스트 23	통과 (12.64ms, 52.9MB)
테스트 8	통과 (12.48ms, 55.5MB)	테스트 24	통과 (14.52ms, 55MB)
테스트 9	통과 (12.51ms, 55.4MB)	테스트 25	통과 (224.18ms, 147MB)
테스트 10	통과 (16.12ms, 55MB)	테스트 26	통과 (238.52ms, 151MB)
테스트 11	통과 (10.69ms, 55.1MB)	테스트 27	통과 (248.04ms, 162MB)
테스트 12	통과 (10.11ms, 52.5MB)	테스트 28	통과 (257.12ms, 162MB)
테스트 13	통과 (11.38ms, 52.9MB)	테스트 29	통과 (252.69ms, 163MB)
테스트 14	통과 (16.43ms, 53.1MB)	테스트 30	통과 (244.93ms, 151MB)
테스트 15	통과 (1.87ms, 52MB)	테스트 31	통과 (270.92ms, 162MB)
테스트 16	통과 (1.99ms, 52.8MB)	테스트 32	통과 (222.10ms, 154MB)

대리님, 코드 리팩토링 완료했습니다...
제가 너무 알고리즘을 얕봤나 보네요.

정확성뿐만 아니라 효율성도 신경쓰는 게
알고리즘의 핵심이랍니다. 기억해 두세요.

Chapter 3

점장... 자넨 유능한 인재야

흠... 그럴 때도 있었지... (아련)

점장, 무슨 생각을 그렇게 하나...
아참, 내가 말한 프로그램은 개발하고 있지?

어... 무슨 프로그램 말씀이십니까?

점장... 내가 배아속의 출퇴근 기록을 위한
관리 페이지를 만들라고 했잖나.

아, 그리고 배아야족들이 가끔 한글 별명을 바꾸고 싶어하니 별명 변경 기능도 넣어주게.

알겠습니다. 사장님
(오픈채팅방이랑 비슷한 걸? 한 번 활용해볼까?)

응용 문제

[응용]

- 기존의 존재한 기능을 활용한 **웹 페이지 제작**
  - Web 구현을 위해 Java 코드를 **JavaScript**로 변환
  - 기존에 리스트로 주어졌던 Record를 직접 입력하는 방식으로 구현
  - 입력 과정에서의 제약 조건 처리를 위해 여러 검증 기능 추가
    (로그인 여부, 입력값의 타당성 등)

JavaScript 코드

```
function solution(record) {
    const idMap = new Map();
    for(let r of record) {
        const split = r.split(" ");
        const cmd = split[0];
        if(cmd != "Leave") {
            idMap.set(split[1], split[2]);
        }
    }
    const answer = [];
    for(let r of record) {
        const split = r.split(" ");
        const cmd = split[0];
        if(cmd != "Change") {
            answer.push(idMap.get(split[1])
                        + (cmd == "Enter" ? "님이 들어왔습니다." : "님이 나갔습니다."))
        }
    }
    return answer;
}
```

Web

Source (html & Css)

```html
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <title>출퇴근</title>
  <style>
    .box {
      float: left;
      border: 1px solid gray;
      padding: 5px;
      margin: 5px;
    }
    input {
      width: 120px;
    }
    #member_table td {
      width: 60px;
    }
    #member_table td:nth-child(1) {
      width: 100px;
    }
    #member_table td:nth-child(2) {
      width: 180px;
    }
  </style>
</head>
```

```html
    <div class="box" style="width: 320px">
      <h3>메시지</h3>
      <div id="message"></div>
    </div>
    <div class="box">
        <h3>멤버 리스트</h3>
        <div id="member_list">
      <table id="member_table">
        <tr>
          <td>ID</td>
          <td>별명</td>
          <td>출근</td>
          <td>퇴근</td>
        </tr>
      </table>
    </div>
    </div>

    <div class="box" id="new_member">
      <h3>멤버 추가</h3>
      <table>
        <tr>
          <td>ID</td>
          <td><input id="new_id" type="text" maxlength="10" minlength="1" /></td>
        </tr>
        <tr>
          <td>별명</td>
          <td><input id="new_nick" type="text" maxlength="10" minlength="1" /></td>
        </tr>
      </table>

      <br />
      <button onclick="add_member()">추가</button>
    </div>
    <script src="record.js"></script>
  </body>
</html>
```

Source (JavaScript)

```
const record = [];
const member = new Map();
const online = new Set();

function message(record) {
  const idMap = new Map();
  for (let r of record) {
    const split = r.split(" ");
    const cmd = split[0];
    if (cmd != "Leave") {
      const id = split[1];
      const nick = split[2];
      idMap.set(id, nick);
    }
  }
  const messages = [];
  for (let r of record) {
    const split = r.split(" ");
    const cmd = split[0];
    if (cmd != "Change") {
      const id = split[1];
      messages.push(
        "<b>" + idMap.get(id) + "</b>" +
          (cmd == "Enter" ? "님이 들어왔습니(
      );
    }
  }
  return messages;
}
```

```
function add_member() {
  const new_id = document.getElementById("new_id").value;
  if(!new RegExp("^[a-zA-Z0-9_]+$").test(new_id)){
    return alert("영문 및 숫자로만 ID를 생성할 수 있습니다!")
  }
  if(member.has(new_id)) {
    alert("이미 존재하는 ID입니다!")
  } else {
    const new_nick = document.getElementById("new_nick").value;
    member.set(new_id, new_nick);
    list_member(member);
  }
}

function list_member(member) {
  let table = `<table id="member_table"><tr><td>ID</td><td>별명</td><td>입장</td><td
  for(let m of member) {
    table += "<tr><td>"+m[0]+"</td><td> \
    <input id=\"nick_"+m[0]+"\" type=\"text\" \
    maxlength=\"10\" minlength=\"1\" value=\""+m[1]+"\"> \
    <button onclick=\"change('"+m[0]+"')\">수정</button></td> \
    <td><button onclick=\"enter('"+m[0]+"')\">출근</button></td> \
    <td><button onclick=\"leave('"+m[0]+"')\">퇴근</button></td></tr>"
  }
  table += "</table>"
  document.getElementById("member_list").innerHTML = table;
}
```

```
function change(id) {
  console.log('수정');
  const change_nick = document.getElementById('nick_'+id).value;
  console.log(id, change_nick);
  member.set(id, change_nick);
  const msg = "Change " + id + " " + change_nick;
  console.log(msg);
  record.push(msg);
  list_member(member);
  list_message(message(record));
}

function enter(id) {
  if(online.has(id)) {
    alert("이미 출근한 ID입니다!")
  } else {
    const msg = "Enter " + id + " " + member.get(id);
    record.push(msg);
    online.add(id);
    list_message(message(record));
  }
}

function leave(id) {
  if(online.has(id)) {
    const msg = "Leave " + id;
    record.push(msg);
    online.delete(id);
    list_message(message(record));
  } else {
    alert("출근하지 않은 ID입니다!")
  }
}

function list_message(messages) {
  document.getElementById("message").innerHTML = "";
  for(let message of messages) {
    document.getElementById("message").innerHTML += message + '<br>';
  }
}
```

Q&A


감사합니다