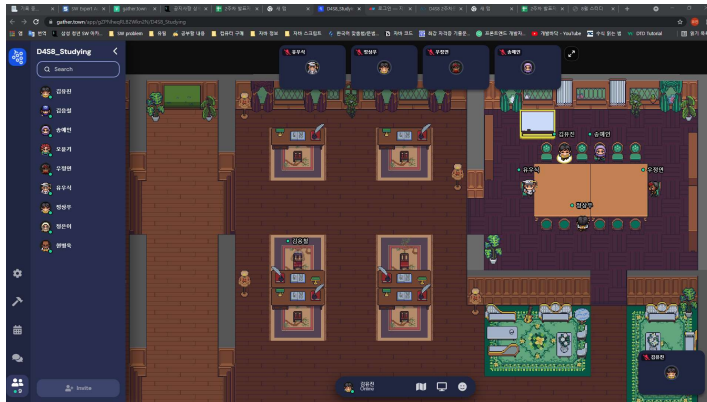
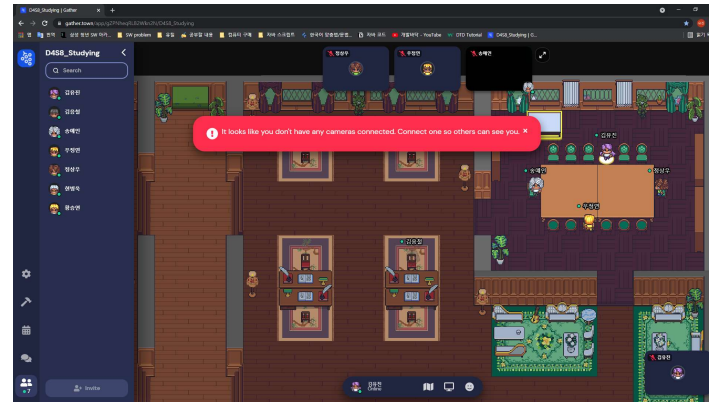


8월 9일 ~ 8월 12일 Gather 모임

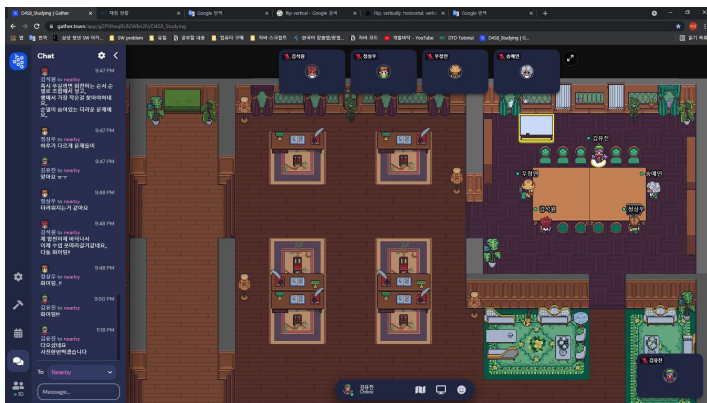
8월 9일



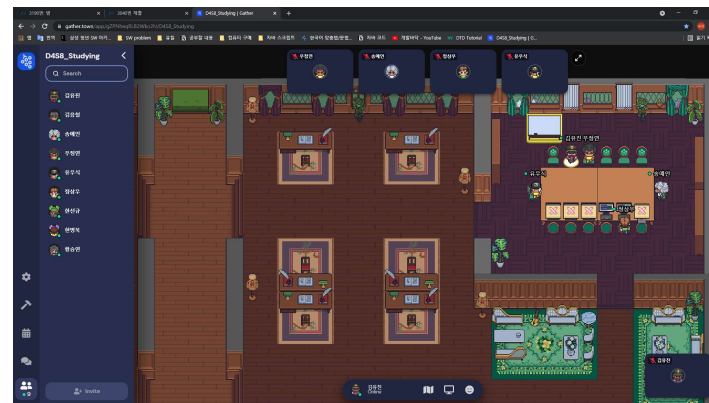
8월 10일



8월 11일



8월 12일



8월 13일 발표화면

The screenshot shows a Google Meet window with a presentation slide. The slide displays a Java program for a linked list. The program includes imports for `BufferedReader`, `FileInputStream`, `IOException`, `InputStreamReader`, `StringTokenizer`, and `StringTokenizer`. It defines a `LinkedList` class with a `map` and a `main` method. The `main` method reads input from a file and processes it to create a linked list.

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedList;
import java.util.StringTokenizer;

public class Baekjoon_1406 {
    static boolean[] map;
    static LinkedList<Snake> s = new LinkedList<>();
    static LinkedList<Apple> a = new LinkedList<>();
    static LinkedList<Command> c = new LinkedList<>();
    static int[] dr = new int[] { 0, 1, 0, -1 }; // 오른쪽으로 가는지 알 수 있음
    static int[] dc = new int[] { 1, 0, -1, 0 };
    static int N, K, L, dir;
    static int time = 0;

    public static void main(String[] args) throws NumberFormatException, IOException {
        System.setIn(new FileInputStream("C:/CodingStudy/Baekjoon/1406/1406_in.txt"));
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;
        N = Integer.parseInt(br.readLine());
        K = Integer.parseInt(br.readLine());
        for (int k = 0; k < K; k++) { // K번의 지령
            st = new StringTokenizer(br.readLine());
            a.add(new Apple(Integer.parseInt(st.nextToken()) - 1, Integer.parseInt(st.nextToken()) - 1)); // 사과 위치
        }
        L = Integer.parseInt(br.readLine());
        for (int l = 0; l < L; l++) { // 지령의 지령
            st = new StringTokenizer(br.readLine());
        }
    }
}
```

The presentation is being shown to four participants in a Google Meet session. The participants are visible in small video windows on the right side of the screen. The bottom of the screen shows the Google Meet interface with a microphone icon, a video icon, and a chat icon. The bottom status bar indicates the time is 9:09 PM and the meeting ID is tzc-eonp-uge.

3190 뱀 : 김유진

- 모든 정보를 객체로 저장
 - 명령어
 - 뱀
 - 사과

```
static class Command {
    int x;
    String com;

    public Command(int x, String com) {
        super();
        this.x = x;
        this.com = com;
    }
}

static class Apple {
    int r;
    int c;

    public Apple(int r, int c) {
        super();
        this.r = r;
        this.c = c;
    }
}

static class Snake {
    int r = 0;
    int c = 0;
    static int len = 1;

    public Snake(int r, int c) {
        super();
        this.r = r;
        this.c = c;
    }
}

static boolean[][] map;
static LinkedList<Snake> s = new LinkedList<>();
static LinkedList<Apple> a = new LinkedList<>();
static LinkedList<Command> c = new LinkedList<>();
static int[] dr = new int[] { 0, 1, 0, -1 }; // 0 1 2 3
```

3190 뱀 : 김유진

1. 시간 증가
2. 수행할 명령이 있는가?
3. 계속 진행 가능한가?

```
time = -1;
s.add(new Snake(0, 0));
do {
    ++time; // 시간 증가 ①
    if (c.size() != 0 && c.peek().x == time) { // 현재 수행할 명령이 있는가?
        switch (c.poll().com) {
            case "D": // 오른쪽이면
                if (++dir >= 4) // 방향 전환
                    dir %= 4;
                break;
            case "L": // 왼쪽이면
                if (--dir < 0) // 방향전환
                    dir = 3;
                break;
        }
    }
    showMap();
} while (!nextState()); // 계속 진행할 수 있는 상황인가? ③
System.out.println(time + 1);
```

3190 뱀 : 김유진

1. 시간 증가
2. 수행할 명령이 있는가?
3. 계속 진행 가능한가?
 1. 맵 확인가?
 2. 내 몸과 충돌했나?
 3. 다음칸에 뭐가 있나?
 1. 사과? - 몸 길이 1 증가
 2. 빈칸
4. 몸을 잘라야 하나?

```
static boolean nextState() { // 다음 상황 확인
```

3-1

```
    int nr = s.get(0).r + dr[dir];  
    int nc = s.get(0).c + dc[dir];  
    if (nr < 0 || nr >= N || nc < 0 || nc >= N) // 맵을 벗어나면 죽음  
        return true;
```

3-2

```
    int cnt = 0;  
    while (++cnt < s.size()) { // 몸이랑 충돌하면 죽음, 머리부터 체크한다.  
        if (nr == s.get(cnt).r && nc == s.get(cnt).c)  
            return true;  
    }
```

3-3

```
    cnt = -1;  
    while (++cnt < a.size()) { // 사과랑 충돌하면 길이가 1 증가함  
        if (nr == a.get(cnt).r && nc == a.get(cnt).c) {  
            ++(s.get(0).len);  
            a.remove(cnt);  
        }  
    }
```

3-4

```
    s.push(new Snake(nr, nc)); // 뱀을 새로운 위치로 집어 넣고  
    if (s.size() > s.get(0).len) // linkedList의 길이가 현재 뱀의 길이보다 길면  
        s.pollLast(); // 꼬리부분 제거 (사과먹으면 제거 안함)
```

```
    return false; // 아무 이상 없다.
```

```
}
```


3190 뱀 : 김유진

추가 디버깅 수행

```
static void showMap() {
    System.out.println();
    System.out.println("time : " + time);
    for (int row = 0; row < N; row++) {
        for (int col = 0; col < N; col++) {
            boolean who = false;
            for (int i = 0; i < s.size(); i++) {
                if (s.get(i).c == col && s.get(i).r == row)
                    who = true;
            }
            if (who)
                System.out.print("x ");
            else
                System.out.print("* ");
        }
        System.out.println();
    }
}
```

```
time : 0
x * * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 1
x x * * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 2
x x x * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 3
* x x x * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 4
* x x x x * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 5
* x x x x x *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 6
* x x x x x x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 7
* * x x x x x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 8
* * * x x x x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 9
* * * * x x x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 10
* * * * * x x
* * * * * x
* * * * * x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 11
* * * * * x x
* * * * * x
* * * * * x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```
time : 12
* * * * * x x
* * * * * x
* * * * * x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

15828 라우터 : 정상우

접근방식

먼저 온 패킷부터 하나씩 처리한 후 버퍼에서 제거한다

1.while문을 통해 입력을 받고 들어온 값을 확인한다.

1-1 값이 0일때 큐를 poll한다.

1-2 값이 0이 아닐때 그 값이 0보다 크고 큐의 사이즈가 N보다 작을때 큐에 offer한다.

종료조건 = 들어온 값이 -1 일때

2 큐가 빌때까지 poll하여 출력한다.

3 큐가 다 비면 empty를 출력한다.

코드

```
package study02;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class Router {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Scanner sc = new Scanner(System.in);

        int N=sc.nextInt();
        int res=0;
        Queue<Integer> q=new LinkedList<Integer>();

        while(res!=-1) {
            res=sc.nextInt();//입력받기
            if(res==0) {
                q.poll();
            }
            else {
                if(res>0&&q.size()<N) {
                    q.offer(res);
                }
            }
        }
    }
}
```

```
    }
}

if(q.isEmpty()) { //큐가 비었있을때 empty 출력
    System.out.println("empty");
}

else { //그렇지 않다면 큐 순서대로출력
    while(!q.isEmpty()) {
        System.out.print(q.poll()+" ");
    }
}

}

}
```

10799 쇠막대기 : 송예인

```
1 package algo;
2
3 import java.io.BufferedReader;
4
5
6
7
8 public class boj10799_v2 {
9     public static void main(String[] args) throws IOException {
10         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11         char[] ch_arr = br.readLine().toCharArray();
12         int cnt = 0, ans = 0;
13         Stack<Character> st = new Stack<>();
14         for(char c: ch_arr) {
15             if(c=='(') {
16                 cnt = 0;
17                 st.push(c);
18                 continue;
19             }
20             cnt++;
21             st.pop();
22             if(c==')') { // 범이면
23                 if(cnt==1) ans+= st.size(); //바로 직전에 (라는 소리
24                 else ans++;
25             }
26         }
27         System.out.println(ans);
28     }
29 }
```

스택활용

- 스택을 이용해서 레이저 전까지의 막대기를 관리한다
- 레이저가 발사되면 잘린 막대기 개수를 더해준다
- 막대기가 끝나면 하나 개수를 더해준다.
- 처음에 boolean[] 을 활용해 레이저의 인덱스 위치에 막대기가 있을 때로 판단하였지만 이중 for문이 되어 통과하지 못하였다

2304 창고다각형 : 우정연

접근방식

1. 가장 높은 기둥 두개를 구함(왼쪽, 오른쪽)
2. 다각형 넓이 구하기
 - 2 - 1. 왼쪽부터 왼쪽 가장 높은 기둥까지 반복 - 자신보다 더 높은 기둥이 나오면 그때까지의 다각형 구함
 - 2 - 2. 오른쪽부터 오른쪽 가장 높은 기둥까지 반복 - 자신보다 더 높은 기둥이 나오면 그때까지의 다각형 구함
 - 2 - 3. 높은 기둥 두 개의 인덱스가 다른 경우 - 그 사이의 다각형을 구함
3. 다각형 넓이 출력

```
public class BJ2304_Warehouse {
    public static void main(String[] args) throws NumberFormatException, IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        int N = Integer.parseInt(in.readLine());
        Pole[] poles = new Pole[N];    // 기둥을 저장할 배열
        StringTokenizer st;
        int l, max = -1, leftMaxIdx = 0, rightMaxIdx = N - 1;
        for(int i = 0 ; i < N ; i++) {    // 기둥을 입력받음
            st = new StringTokenizer(in.readLine());
            poles[i] = new Pole(Integer.parseInt(st.nextToken()) - 1, Integer.parseInt(st.nextToken()));
        }
        Arrays.sort(poles); // poles를 인덱스 기준으로 정렬함
        for(int i = 0 ; i < N ; i++) {    // 왼쪽 가장 높은 기둥의 인덱스(기둥 배열에서의 인덱스)를 저장
            if(poles[i].height == max) {
                leftMaxIdx = i;
                break;
            }
        }
        for(int i = N - 1 ; i >= 0 ; i--) {
            if(poles[i].height == max) {    // 오른쪽 가장 높은 기둥의 인덱스를 저장
                rightMaxIdx = i;
                break;
            }
        }
    }
}
```

2304 창고다각형 : 우정연

```
int area = 0;          // 다각형 면적
int lastIdx = 0;       // 이전 인덱스
for(int i = 1 ; i <= leftMaxIdx ; i++) {    // 왼쪽부터 왼쪽 가장 높은 기둥까지, poles 배열의 인덱스
    if(poles[lastIdx].height <= poles[i].height) {    // 더 높은 기둥을 찾으면
        area += poles[lastIdx].height * (poles[i].idx - poles[lastIdx].idx); // 전 기둥부터 지금 바로 전 기둥까지의 다각형 면적 더함
        lastIdx = i;    // 이번 기둥 인덱스를 이전으로 저장
    }
}
lastIdx = N - 1;
int a = 0;
for(int i = N - 2 ; i >= rightMaxIdx ; i--) {    // 오른쪽부터 오른쪽 가장 높은 기둥까지
    if(poles[lastIdx].height <= poles[i].height) {    // 더 높은 기둥을 찾으면
        area += poles[lastIdx].height * (poles[lastIdx].idx - poles[i].idx); // 전 기둥부터 지금 바로 전 기둥까지의 다각형 면적 더함
        lastIdx = i;    // 이번 기둥 인덱스를 이전으로 저장
    }
}
area += poles[leftMaxIdx].height;    // 가장 높은 기둥을 더함
if(leftMaxIdx != rightMaxIdx) {    // 가장 높은 기둥이 여러개이면
    area += poles[leftMaxIdx].height * (poles[rightMaxIdx].idx - poles[leftMaxIdx].idx);    // 그 사이 면적 더함
}
System.out.println(area);    // 결과를 출력
}

static class Pole implements Comparable<Pole>{    // 기둥 클래스
    int idx;    // 인덱스
    int height;    // 높이
    public Pole(int idx, int height) {
        this.idx = idx;
        this.height = height;
    }
    @Override
    public int compareTo(Pole o) {    // 인덱스로 정렬하기 위해 compareTo 메소드 구현
        int sub = this.idx - o.idx;
        if(sub > 0) return 1;
        else if(sub < 0) return -1;
        else return 0;
    }
}
}
```