IM 어려운 문제 백준<줄 세우기, 빙고, 경비원, 일곱 난쟁이>

문제를 풀면서 사소한 실수들을 디버깅 하는 작업이 많았는데 계획을 철저하게 세우고 코드작업에 들어가야 실수를 줄일 수 있다는 것을 다시 한번 느꼈고, 코드가 길어질 것 같은 문제들도 의심을 갖지 않고 차근차근 작성해 나가면 맞다는 것을 알 수 있었습니다.

```java
package day0827;

import java.io.BufferedReader;

public class BaekJoon_줄_세우기 {

    public static void main(String[] args) throws NumberFormatException, IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        ArrayList<Integer> orderList = new ArrayList<Integer>();
        int N = Integer.parseInt(br.readLine());
        StringTokenizer st = new StringTokenizer(br.readLine());
        int order[] = new int[N+1];
        for(int i=0; i<N; i++) {
            order[i] = Integer.parseInt(st.nextToken());
        }
        for(int i=0; i<N; i++) {

        orderList.add(i-order[i], i+1);
        }

        for(int i=0; i<N; i++) {
            System.out.print(orderList.get(i)+" ");
        }
    }
}
```

```java
import java.io.BufferedReader;

public class BaekJoon_경비원 { // 1,2,3,4 - 북,남,서,동 순서

    static int N, M;
    static Point[] point;

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        N = Integer.parseInt(st.nextToken());
        M = Integer.parseInt(st.nextToken());
        int minSum = 0;

        int G = Integer.parseInt(br.readLine());
        point = new Point[G + 1];
        for (int i = 0; i <= G; i++) {
            st = new StringTokenizer(br.readLine());
            int a = Integer.parseInt(st.nextToken());
            int b = Integer.parseInt(st.nextToken());
            if (a == 1) { // 북
                point[i] = new Point(a, 0, b);
            }
            if (a == 2) { // 남
                point[i] = new Point(a, M, b);
            }
            if (a == 3) { // 서
                point[i] = new Point(a, b, 0);
            }
            if (a == 4) { // 동
                point[i] = new Point(a, b, N);
            }
        }
//      System.out.println(Arrays.toString(point));

        for (int i = 0; i < G; i++) {
            if (point[G].d == 1 && point[i].d == 2 || point[G].d == 2 && point[i].d == 1) { // 주인공과 친구가 남-북 or 북-남 일 때
                minSum += Math.min(point[G].y + M + point[i].y, N - point[G].y + M + N - point[i].y);

            } else if (point[G].d == 3 && point[i].d == 4 || point[G].d == 4 && point[i].d == 3) { // 주인공과 친구가 서-동 or 동-서 일 때
                minSum += Math.min(point[G].x + point[i].x + N, M - point[G].x + N + M - point[i].x);

            } else { // 주인공과 친구가 나머지일 때
                minSum += (Math.abs(point[G].x - point[i].x) + Math.abs(point[G].y - point[i].y));
            }
        }

        System.out.println(minSum);

    }

static class Point {
    int d;
    int x;
    int y;

    public Point(int d, int x, int y) {
        this.d = d;
        this.x = x;
        this.y = y;
    }

    @Override
    public String toString() {
        return "Point [d=" + d + ", x=" + x + ", y=" + y + "]";
    }

}
```

```java
import java.io.BufferedReader;

public class BaekJoon_일곱_난쟁이 {

    static int arr[];
    static boolean used[];
    static ArrayList<Integer> finalArr;

    public static void main(String[] args) throws NumberFormatException, IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        arr = new int[9];
        finalArr = new ArrayList<Integer>();
        for (int i = 0; i < 9; i++) {
            arr[i] = Integer.parseInt(br.readLine());
        }
        Arrays.sort(arr);
        used = new boolean[9];
        comb(0, 0);
        for (int i = 0; i < 7; i++) {
            if (finalArr.get(i) != 0) {
                System.out.println(finalArr.get(i)); // true가 된 애들이 차곡차곡 저장되어 있기때문에 처음 들어왔던 7개만 출력해준다.
            }
        }

    }

    static void comb(int target, int cnt) {
        if (cnt == 7) {
            int sum = 0;
            for (int i = 0; i < used.length; i++) {
                if (used[i]) {
                    sum += arr[i];
                }
            }
            if (sum == 100) {
                for (int i = 0; i < used.length; i++) {
                    if (used[i]) {
//                      System.out.println(arr[i]); // 100이 되는 모든 쌍을 찾으면 안되고 한쌍을 찾는 순간 끝내야한다. 따라서 배열을 하나 만들어서 최종으로 저장해줄거다.

                        finalArr.add(arr[i]);
                    }
                }
            }
            return;

        }
        if (target > 8) {
            return;
        }

        used[target] = true;
        comb(target + 1, cnt + 1);
        used[target] = false;
        comb(target + 1, cnt);

    }
}
```

```java
package day0827;

import java.io.BufferedReader;

public class BaekJoon_빙고 {

    static int map[][];
    static int bingoCnt;
    static int numCnt;

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        map = new int[5][5];

        for (int i = 0; i < 5; i++) {
            StringTokenizer st = new StringTokenizer(br.readLine());
            for (int j = 0; j < 5; j++) {
                map[i][j] = Integer.parseInt(st.nextToken());
            }
        }

        Loop: for (int i = 0; i < 5; i++) {
            StringTokenizer st = new StringTokenizer(br.readLine());
            for (int j = 0; j < 5; j++) {
                numCnt++;
                delete(Integer.parseInt(st.nextToken())); // 입력값 하나 하나 삭제하면서 함수로 전체 검색
                rowBingo();
                columnBingo();
                rightDiagonalBingo();
                leftDiagonalBingo();
                if (bingoCnt >= 3) {
                    System.out.println(numCnt);
                    break Loop;
                }

                bingoCnt = 0; // 빙고의 숫자가 중복되지 않게 탐색이 끝나면 0으로 초기화

            }
        }
    }

    static void delete(int a) {
        Loop: for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (map[i][j] == a) {
                    map[i][j] = 0;
                    break Loop;
                }
            }
        }
    }

    static void rowBingo() {
        int cnt = 0;
        for (int i = 0; i < 5; i++) {
            cnt = 0;
            for (int j = 0; j < 5; j++) {
                if (map[i][j] == 0) {
                    cnt++;
                }
                if (cnt == 5) {
                    bingoCnt++;
                }
            }
        }
    }

    static void columnBingo() {
        int cnt = 0;
        for (int j = 0; j < 5; j++) {
            cnt = 0;
            for (int i = 0; i < 5; i++) {
                if (map[i][j] == 0) {
                    cnt++;
                }
                if (cnt == 5) {
                    bingoCnt++;
                }
            }
        }
    }

    static void rightDiagonalBingo() {
        int cnt = 0;
        int a = 0;
        for (int i = 0; i < 5; i++) {
            if (map[a][4 - a] == 0) {
                cnt++;
            }
            a++;
        }
        if (cnt == 5) {
            bingoCnt++;
        }
    }

    static void leftDiagonalBingo() {
        int cnt = 0;
        int a = 0;
        for (int i = 0; i < 5; i++) {
            if (map[a][a] == 0) {
                cnt++;
            }
            a++;
        }
        if (cnt == 5) {
            bingoCnt++;
        }
    }
}
```