1. "사다리 타기" 정답코드

```
class Solution {
        public char[] solution(int n, int[][] ladder){
                char[] answer = new char[n];
                for(int i = 0; i < n; i++){
                        answer[i] = (char)(i + 65);
                }
                for(int[] line : ladder){
                        for(int x : line){
                                char tmp = answer[x];
                                answer[x] = answer[x-1];
                                answer[x-1] = tmp;
                        }
                }
                return answer;
        }
}
```

▶▶ 코드 해설

간단한 시뮬레이션(일정한 명령에 따라 개체를 이동시키는 알고리즘)으로 해결할 수 있습니다. 일단 answer 리스트에 각 학생의 알파벳을 순서대로 담습니다. 그리고 사다리의 가로줄을 위에서부터 차례대로 탐색합니다.

첫 번째 가로줄에서 가로막대가 발견되면 해당 가로막대가 연결하고 있는 두 세로줄의 알파벳을 교환합니다. 이런 과정을 첫 번째 가로줄의 모든 가로막대에서 하게되면 모든 학생이 첫 번째 가로줄을 사다리 탄 결과가 answer에 표현됩니다.

위에 과정을 마지막 가로줄까지 처리하면 answer에 최종결과가 담겨져 있습니다.

2. "청소" 정답코드

```
class Solution {
        public int[] solution(int[][] board, int k){
                int[] answer = new int[2];
                int n = board.length;
                int[] dx = {-1, 0, 1, 0};
                int[] dy = {0, 1, 0, -1};
                int x = 0, y = 0, d = 1, count = 0;
                while(count < k){
                        count++;
                        int nx = x + dx[d];
                        int ny = y + dy[d];
                        if(nx < 0 || nx >= n || ny < 0 || ny >= n || board[nx][ny] == 1){
                                d = (d + 1) % 4;
                                continue;
                        }
                        x = nx;
                        y = ny;
                }
                answer[0] = x;
                answer[1] = y;
                return answer;
        }
}
```

▶▶ Comment :

위에 d 변수는 로봇이 지도 끝을 만나거나 장애물을 만났을 때 1증가시킨다.
d변수가 1증가는 것은 로봇이 오른쪽으로 90도 회전하는 효과가 일어난다.
만약 d값이 1이면 로봇이 3시방향으로 이동하는데 d값이 2가 되면 로봇이 6시 방향으로 이동하게 된다. 또 d값이 3이면 9시 방향이고, 1증가 시키는 공식 d = (3+1) % 4 을 하면 다시 d값이 0이 되어 12시 방향을 보게 된다.

3. "잃어버린 강아지" 정답코드

```
class Solution {
        public int solution(int[][] board){
                int n = board.length;
                int[] dx = {-1, 0, 1, 0};
                int[] dy = {0, 1, 0, -1};
                int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
                for(int i = 0; i < n; i++){
                        for(int j = 0; j < n; j++){
                                if(board[i][j] == 2){
                                        x1 = i;
                                        y1 = j;
                                }
                                if(board[i][j] == 3){
                                        x2 = i;
                                        y2 = j;
                                }
                        }
                }
                int d1 = 0, d2 = 0, count = 0;
                while(count < 10000){
                        count++;
                        int nx1 = x1 + dx[d1];
                        int ny1 = y1 + dy[d1];
                        int nx2 = x2 + dx[d2];
                        int ny2 = y2 + dy[d2];
                        boolean flag1 = true, flag2 = true;
                        if(nx1 < 0 || nx1 >= n || ny1 < 0 || ny1 >= n || board[nx1][ny1] == 1){
                                d1 = (d1 + 1) % 4;
                                flag1 = false;
                        }
                        if(nx2 < 0 || nx2 >= n || ny2 < 0 || ny2 >= n || board[nx2][ny2] == 1){
                                d2 = (d2 + 1) % 4;
                                flag2 = false;
                        }
                        if(flag1 == true){
                                x1 = nx1;
                                y1 = ny1;
                        }
                        if(flag2 == true){
                                x2 = nx2;
                                y2 = ny2;
                        }
                        if(x1 == x2 && y1 == y2) break;
                }
                if(count >= 10000) return 0;
                return count;
        }
}
```

▶▶ Comment :

flag1, flag2 변수를 두어 현수와 강아지를 현재방향으로 이동할지를 결정합니다.
만약 현수가 회전을 했다면 flag1은 false가 되어

```
if(flag1 == true){
        x1 = nx1;
        y1 = ny1;
}
```

위에 if 조건문이 참이 되지 않아 nx1, ny1 지점으로 이동을 하지 않고 이 순간에는 회전만 하게 됩니다.

4. "좌석번호" 정답코드

```
class Solution {
        public int[] solution(int c, int r, int k){
                int[] answer = new int[2];
                if(k > c * r) return new int[] {0, 0};
                int[][] seat = new int[c][r];
                int[] dx = {-1, 0, 1, 0};
                int[] dy = {0, 1, 0, -1};
                int x = 0, y = 0, count = 1, d = 1;
                while(count < k){
                        int nx = x + dx[d];
                        int ny = y + dy[d];
                        if(nx < 0 || nx >= c || ny < 0 || ny >=r || seat[nx][ny] > 0){
                                d = (d + 1) % 4;
                                continue;
                        }
                        seat[x][y] = count;
                        count++;
                        x = nx;
                        y = ny;
                }
                answer[0] = x + 1;
                answer[1] = y + 1;
                return answer;
        }
}
```

▶▶ Comment :

```
int[][] seat = new int[c][r];
```

강연장을 시계방향으로 90도 회전했다고 가정하기 때문에 2차원 배열의 행의 크기(세로길이)가 c가 되고 2차원 배열의 열의 크기(가로길이)가 r이 됩니다.

```
if(nx < 0 || nx >= c || ny < 0 || ny >=r || seat[nx][ny] > 0){
        d = (d + 1) % 4;
        continue;
}
```

앉으려는 좌석이 강연장 밖으로 나가거나, 이미 앉으려는 좌석이 사람이 앉아 있다면 좌석 배치 방향을 시계방향으로 90도 회전합니다. 회전만 했지 사람이 앉지는 않았으므로 continue해서 좌석배치와 count 증가를 건너뜁니다.

5. "최대길이 바이토닉 수열" 정답코드

```
class Solution {
        public int solution(int[] nums){
                int answer = 0;
                int n = nums.length;
                ArrayList<Integer> peaks = new ArrayList<>();
                for(int i = 1; i < n-1; i++){
                        if(nums[i-1] < nums[i] && nums[i] > nums[i+1]){
                                peaks.add(i);
                        }
                }
                for(int x : peaks){
                        int left = x;
                        int right = x;
                        int cnt = 1;
                        while(left-1 >= 0 && nums[left-1] < nums[left]){
                                left--;
                                cnt++;
                        }
                        while(right+1 < n && nums[right] > nums[right+1]){
                                right++;
                                cnt++;
                        }
                        answer = Math.max(answer, cnt);
                }
                return answer;
        }
}
```

▶▶ Comment :

수열에는 여러개의 바이토닉 수열이 존재할 수 있습니다.
각각의 바이토닉 수열의 봉우리 지점의 인덱스번호를 peaks 배열에 저장합니다.
peaks를 탐색하면서 각각의 바이토닉 수열의 봉우리 지점에서 왼쪽으로 감소수열이 끝날 때 까지 탐색, 오른쪽으로 감소수열이 끝날 때 까지 탐색하면 됩니다. 각 바이토닉 수열의 길이는 봉우리부터 왼쪽으로 감소수열을 탐색하고, 오른쪽으로 감소수열을 탐색한 총 탐색횟수가 해당 바이토닉수열의 길이가 됩니다.

6. "과일 가져가기" 정답코드

```
class Solution {
        public int getMin(int[] fruit){
                int min = 100;
                for(int x : fruit){
                        min = Math.min(min, x);
                }
                return min;
        }
        public Boolean isMinUnique(int[] fruit){
                int cnt = 0;
                int min = getMin(fruit);
                for(int x : fruit){
                        if(x == min) cnt++;
                }
                return cnt == 1;
        }
        public int getMinIndex(int[] fruit){
                int min = getMin(fruit);
                for(int i = 0; i < 3; i++){
                        if(fruit[i] == min) return i;
                }
                return 0;
        }
        public int solution(int[][] fruit){
            int answer = 0;
            int n = fruit.length;
            int[] ch = new int[n];
            for(int i = 0; i < n; i++){
                    if(ch[i] == 1) continue;
                    if(isMinUnique(fruit[i]) == false) continue;
                    for(int j = i+1; j < n; j++){
                            if(ch[j] == 1) continue;
                            if(isMinUnique(fruit[j]) == false) continue;
                            int a = getMinIndex(fruit[i]);
                            int b = getMinIndex(fruit[j]);
                            if(a != b && fruit[i][b] > 0 && fruit[j][a] > 0){
                                  if(fruit[i][a] + 1 <= fruit[i][b] - 1 && fruit[j][b] + 1 <= fruit[j][a] - 1){
                                                fruit[i][a]++;
                                                fruit[i][b]--;
                                                fruit[j][b]++;
                                                fruit[j][a]--;
                                                ch[i] = 1;
                                                ch[j] = 1;
                                                break;
                                        }
                                }
                        }
                }
            }
            for(int[] x : fruit){
                  answer += getMin(x);
            }
            return answer;
        }
}
```

▶▶ Comment :

```
if(a != b && fruit[i][b] > 0 && fruit[j][a] > 0)
```

서로 교환하려하는 과일이 같은 과일이면 교환해도 변화가 전혀 없기 때문에
a != b를 만족해야 합니다. 또한 i번 학생이 주려고 하는 과일 fruit[i][b]이 0개는 아니어야 하고, j번 학생이 주려고 하는 과일 fruit[j][a]이 0개는 아니어야 합니다.

```
if(fruit[i][a] + 1 <= fruit[i][b] - 1 && fruit[j][b] + 1 <= fruit[j][a] - 1)
```

i번 학생이 한 개의 과일을 받은 바구니의 개수 fruit[i][a] + 1 가 i번 학새이 한 개를 준 바구니의 개수 fruit[i][b] - 1 보다 작거나 같아야 과일을 바꾸었을 때 i번 학생이 이득을 봅니다. j번 학생의 fruit[j][b] + 1 <= fruit[j][a] - 1 코드도 i번 학생과 같은 논리입니다.

7. "비밀번호" 정답코드

```
class Solution {
        public int solution(int[] keypad, String password){
                int answer = 0;
                int[] dx = {-1, -1, 0, 1, 1, 1, 0, -1};
                int[] dy = {0, 1, 1, 1, 0, -1, -1, -1};
                int[][] pad = new int[3][3];
                int[][] dist = new int[10][10];
                for(int i = 0; i < 3; i++){
                        for(int j = 0; j < 3; j++){
                                pad[i][j] = keypad[i*3 + j];
                        }
                }
                for(int i = 0; i < 10; i++){
                        Arrays.fill(dist[i], 2);
                }
                for(int i = 0; i < 10; i++) dist[i][i] = 0;
                for(int i = 0; i < 3; i++){
                        for(int j = 0; j < 3; j++){
                            int from = pad[i][j];
                            for(int k = 0; k < 8; k++){
                                if(i+dx[k] >= 0 && i+dx[k] < 3 && j+dy[k] >= 0 && j+dy[k] < 3){
                                        int to = pad[i+dx[k]][j+dy[k]];
                                        dist[from][to] = 1;
                                }
                            }
                        }
                }
                for(int i = 1; i < password.length(); i++){
                        int from = (int)password.charAt(i-1)-48;
                        int to = (int)password.charAt(i)-48;
                        answer += dist[from][to];
                }
                return answer;
        }
}
```

▶▶ Comment :

```
for(int i = 0; i < 10; i++) Arrays.fill(dist[i], 2);
```

키패드에 있는 숫자들 간의 최대 이동시간은 2초입니다. 2초보다 더 긴 이동시간은 없습니다.

8. "회의실 만남" 정답코드

```
class Solution {
        public int[] solution(int[] enter, int[] exit){
                int n = enter.length;
                for(int i = 0; i < n; i++){
                        enter[i]--;
                        exit[i]--;
                }
                int[] enterIdx = new int[n];
                for(int i = 0; i < n; i++){
                        enterIdx[enter[i]] = i;
                }
                int[] enterT = new int[n];
                int[] exitT = new int[n];
                int cnt = 0;
                for(int i = 0, j = 0; i < n; i++){
                        while(j < n && j <= enterIdx[exit[i]]){
                                enterT[enter[j]] = cnt++;
                                j++;
                        }
                        exitT[exit[i]] = cnt++;
                }
                int[] answer = new int[n];
                for(int i = 0; i < n; i++){
                        for(int j = i + 1; j < n; j++){
                                if(!(exitT[i] < enterT[j] || exitT[j] < enterT[i])){
                                        answer[i]++;
                                        answer[j]++;
                                }
                        }
                }
                return answer;
        }
}
```

▶▶ Comment :

```
for(int i = 0; i < n; i++) enterIdx[enter[i]] = i;
```

각 사람이 몇 번째로 회의실에 들어왔는지 쉽게 알기 위해 enterIdx에 각 사람의 입실 번째를 기록합니다. 만약 enter가 [1, 4, 2, 3] 이라면 먼저 [0, 3, 1, 2]로 하나씩 작아지고 난 후 위에 코드를 하게 되는데 enterIdx[0]값은 0, enterIdx[3]값은 1이 되는데 의미는 0번 사람은 회이실에 0번째로 입실했고, 3번 사람은 1번째로 회의실에 입실했다는 의미입니다.