1. "최대 길이 연속수열" 정답코드

```
class Solution {
        public int solution(int[] nums){
                int answer = 0;
                HashSet<Integer> set = new HashSet<>();
                for(int x : nums) set.add(x);
                for(int x : set){
                        if(set.contains(x - 1)) continue;
                        int cnt = 0;
                        while(set.contains(x)){
                                cnt++;
                                x++;
                        }
                        answer = Math.max(answer, cnt);
                }
                return answer;
        }
}
```

▶▶ Comment :

수열에 자기보다 1작은 숫자가 없다면 자기 자신이 연속수열의 시작 숫자가 되어 1씩 증가하면서 연속수열인지 계속 확인해 갑니다.

2. "문자열 압축해제" 정답코드

```
class Solution {
    public String solution(String s){
        String answer = "";
        Stack<String> st = new Stack<>();
        for(Character x : s.toCharArray()){
            if(x == ')'){
                String tmp = "";
                while(!st.empty()){
                    String c = st.pop();
                    if(c.equals("(")){
                        String num = "";
                        while(!st.empty() && Character.isDigit(st.peek().charAt(0))){
                            num = st.pop() + num;
                        }
                        String res = "";
                        int cnt = 0;
                        if(num.equals("")) cnt = 1;
                        else cnt = Integer.parseInt(num);
                        for(int i = 0; i < cnt; i++) res += tmp;
                        st.push(res);
                        break;
                    }
                    tmp = c + tmp;
                }
            }
            else st.push(String.valueOf(x));
        }
        for(String x : st) answer += x;
        return answer;
    }
}
```

▶▶ Comment :

이 문제는 스택에 문자열을 저장한다는 아이디어가 중요합니다.

3. "현관문 출입 순서" 정답코드

```
class Solution {
        public int[] solution(int[] arrival, int[] state){
                Queue<Integer> enter = new LinkedList<>();
                Queue<Integer> exit = new LinkedList<>();
                int n = arrival.length, prev = 1;
                int[] answer = new int[n];
                for(int t = 0, i = 0, cnt = 0; ; t++){
                if(enter.isEmpty() && exit.isEmpty() && i < n) {
                                if(t < arrival[i]){
                                        t = arrival[i];
                                        prev = 1;
                                }
                }
                while(i < n && arrival[i] <= t) {
                                if (state[i] == 0) enter.offer(i);
                                else exit.offer(i);
                        i++;
                        }
                if(prev == 1) {
                                if(!exit.isEmpty()) {
                                        answer[exit.poll()] = t;
                                        prev = 1;
                        }
                                else{
                                        answer[enter.poll()] = t;
                                        prev = 0;
                                }
                }else if(prev == 0) {
                                if(!enter.isEmpty()) {
                                        answer[enter.poll()] = t;
                                        prev = 0;
                                }else{
                                        answer[exit.poll()] = t;
                                         prev = 1;
                                }
                }
                cnt++;
                if(cnt == n) break;
                }
                return answer;
        }
}
```

▸▸ Comment :

두 개의 큐가 비어있고 현재 시간이 다음 도착할 사원의 도착시간보다 작으면 현재 시간을 다음 도착할 사원의 시간으로 바꿔버리는 코드가 중요합니다.

그래야 그 다음 while(i < n && arrival[i] <= t)문에서 큐를 채웁니다.

4. "피부과" 정답코드

```
class Solution {
        public int getTime(String time){
                int H = Integer.parseInt(time.split(":")[0]);
                int M = Integer.parseInt(time.split(":")[1]);
                return H*60+M;
        }
        public int solution(int[] laser, String[] enter){
                int answer = 0;
                int n = enter.length;
                int[][] inList = new int[n][2];
                for(int i = 0; i < n; i++){
                        int a = getTime(enter[i].split(" ")[0]);
                        int b = Integer.parseInt(enter[i].split(" ")[1]);
                        inList[i][0] = a;
                        inList[i][1] = b;
                }
                Queue<Integer> Q = new LinkedList<>();
                Q.offer(inList[0][1]);
                int fT = inList[0][0];
                int pos = 1;
                for(int t = fT; t <= 1200; t++){
                        if(pos < n && t == inList[pos][0]){
                                if(Q.isEmpty() && t > fT) fT= t;
                                Q.offer(inList[pos][1]);
                                pos++;
                        }
                        if(t == fT && !Q.isEmpty()){
                                int idx = Q.poll();
                                fT += laser[idx];
                        }
                        answer = Math.max(answer, Q.size());
                }
                return answer;
        }
}
```

▶▶ Comment :

변수 fT는 레이저실에서 치료받고 있는 고객의 치료가 끝나는 시간입니다.
Q자료구조는 피부과의 대기실로 생각하면 됩니다.

5. "cpu 스케쥴링" 정답코드

```
class Solution {
    public int[] solution(int[][] tasks){
        int n = tasks.length;
        int[] answer = new int[n];
        LinkedList<int[]> programs = new LinkedList<>();
        for(int i = 0; i < n; i++){
            programs.add(new int[]{tasks[i][0], tasks[i][1], i});
        }
        programs.sort((a, b) -> a[0] - b[0]);
        PriorityQueue<int[]> pq = new PriorityQueue<>((a, b) -> a[0] == b[0] ? a[1] - b[1] : a[0] -
b[0]);
        int fT = 0, idx = 0;
        while(!programs.isEmpty() || !pq.isEmpty()){
            if(pq.isEmpty()) fT = Math.max(fT, programs.peek()[0]);
            while(!programs.isEmpty() && programs.peek()[0] <= fT){
                    int[] x = programs.pollFirst();
                    pq.add(new int[]{x[1], x[2]});
            }
            int[] ex = pq.poll();
            fT = fT + ex[0];
            answer[idx++] = ex[1];
        }
        return answer;
    }
}
```

▶▶ Comment :

변수 fT는 cpu가 현재 진행하고 있는 작업이 끝나는 시간입니다.

```
if(pq.isEmpty()) fT = Math.max(fT, programs.peek()[0]);
while(!programs.isEmpty() && programs.peek()[0] <= fT){
        int[] x = programs.pollFirst();
        pq.add(new int[]{x[1], x[2]});
}
```

첫 작업이 처리되는 시점에 if(pq.isEmpty()) fT = Math.max(fT, programs.peek()[0]); 코드가 실행되어 fT값은 첫 작업의 호출시간이 되고, while문에서 첫 작업은 우선순위큐로 들어가 대기하는 작업으로 처리됩니다. 물론 while문은 fT보다 호출시간이 작은 모든 작업을 대기처리하는 코드입니다.

6. "가장 많이 사용된 회의실" 정답코드

```
class Solution {
    public int solution(int n, int[][] meetings){
        int answer = 0;
        int[] res = new int[n];
        PriorityQueue<int[]> ends = new PriorityQueue<>((a, b) -> a[0] == b[0] ? a[1] - b[1] : a[0] -
b[0]);
        TreeSet<Integer> rooms = new TreeSet<>();
        for(int i = 0; i < n; i++) rooms.add(i);
        Arrays.sort(meetings, (a, b) -> a[0] - b[0]);
        for(int[] m : meetings){
            while(!ends.isEmpty() && ends.peek()[0] <= m[0]) rooms.add(ends.poll()[1]);
            if(!rooms.isEmpty()){
                int room = rooms.pollFirst();
                res[room]++;
                ends.add(new int[]{m[1], room});
            }
            else{
                int[] e = ends.poll();
                res[e[1]]++;
                ends.add(new int[]{e[0] + (m[1] - m[0]), e[1]});
            }
        }
        int maxi = 0;
        for(int i = 0; i < n; i++){
            if(res[i] > maxi){
                maxi = res[i];
                answer = i;
            }
        }
        return answer;
    }
}
```

▶▶ Comment :

TreeSet 자료구조는 자료를 추가하거나 자료를 제거할 때 O(log n)으로 오름차순 자동정렬을 해줍니다.

```
if(!rooms.isEmpty()){     }
else{          }
```

if(!rooms.isEmpty()) 구문은 비어있는 회의실이 있으면 작업을 비어있는 회의실 중 번호가 가장 작은 회의실에 할당합니다.

else{ } 는 비어있는 회의실이 없을 때 진행중인 회의실 중 가장 빨리 끝나는 회의실을 현재 작업에 할당합니다.