1. "이진수 정렬" 정답코드

```
class Solution {
        public int[] solution(int[] nums){
                int[] answer = new int[nums.length];
                int[][] res = new int[nums.length][2];
                for(int i = 0; i < nums.length; i++){
                        int cnt = 0;
                        int tmp = nums[i];
                        while(tmp > 0){
                                cnt += (tmp % 2);
                                tmp = tmp / 2;
                        }
                        res[i][0] = nums[i];
                        res[i][1] = cnt;
                }
                Arrays.sort(res, (a, b) -> a[1] == b[1] ? a[0] - b[0] : a[1] - b[1]);
                for(int i = 0; i < res.length; i++){
                        answer[i] = res[i][0];
                }
                return answer;
        }
}
```

▶▶ Comment :

```
while(tmp > 0){
        cnt += (tmp % 2);
        tmp = tmp / 2;
}
```

위 코드는 십진수 tmp를 이진수화 하는 코드입니다. 특히 cnt에 2로 나눈 나머지값을 누적해서 더해주면 0과 1를 더하는데 그렇게 하면 cnt는 최종적으로 1의 개수를 의미합니다.

2. "수열 찾기" 정답코드

```
class Solution {
        public int[] solution(int[] nums){
                int[] answer = new int[nums.length / 2];
                HashMap<Integer, Integer> nH = new HashMap<>();
                for(int x : nums) nH.put(x, nH.getOrDefault(x, 0) + 1);
                Arrays.sort(nums);
                int idx = 0;
                for(int x : nums){
                        if(nH.get(x) == 0) continue;
                        answer[idx] = x;
                        idx++;
                        nH.put(x, nH.get(x) - 1);
                        nH.put(x*2, nH.get(x*2) - 1);
                }
                return answer;
        }
}
```

3. "카드가져가기" 정답코드

```
class Solution {
        public int solution(int[] nums, int k){
                int answer = 0;
                int n = nums.length;
                Integer[] tmp = Arrays.stream(nums).boxed().toArray(Integer[]::new);
                Arrays.sort(tmp, (a, b) -> b - a);
                Integer[] diff = new Integer[n/2];
                for(int i = 0; i < n/2; i++){
                        answer += tmp[i*2+1];
                        diff[i] = tmp[i*2]-tmp[i*2+1];
                }
                Arrays.sort(diff,(a, b) -> b - a);
                for(int i = 0; i < k; i++){
                        answer += diff[i];
                }
                return answer;
        }
}
```

▶▶ Comment :

정수형 배열의 내림차순은 기본형인 int형은 안되고, 클래스 형인 Integer형으로 해야 됩니다.

4. "심사위원" 정답코드

```
class Solution {
        public int getAve(int[] score, int s, int e){
                int sum = 0;
                for(int i = s; i <= e; i++){
                        sum += score[i];
                }
                return (int)Math.floor((sum / (e - s + 1)));
        }
        public int solution(int[] score, int k){
                int n = score.length;
                Arrays.sort(score);
                for(int i = 0; i <= n - k; i++){
                        if(score[i + k - 1] - score[i] <= 10)
                                return getAve(score, i, i + k - 1);
                }
                return 0;
        }
}
```

▶▶ Comment :

```
if(score[i + k - 1] - score[i] <= 10)
        return getAve(score, i, i + k - 1);
```

길이가 k인 연속구간의 가장 큰 숫자와 가장 작은 숫자의 차가 10이하인 구간이 최초로 발견되는 해당 구간의 평균이 답입니다.

5. "모임장소" 정답코드

```
class Solution {
        public int solution(int[][] board){
                int answer=0;
                int n = board.length;
                ArrayList<Integer> row = new ArrayList<>();
                ArrayList<Integer> col = new ArrayList<>();
                for(int i = 0; i < n; i++){
                        for(int j = 0; j < n; j++){
                                if(board[i][j] == 1){
                                        row.add(i);
                                        col.add(j);
                                }
                        }
                }
                col.sort((a, b) -> a - b);
                int x = row.get(row.size() / 2);
                int y = col.get(col.size() / 2);
                for(int p : row) answer += Math.abs(x - p);
                for(int p : col) answer += Math.abs(y - p);
                return answer;
        }
}
```

▶▶ Comment :

```
row.sort((a, b) -> a - b);
```

위 코드는 굳이 할 필요가 없습니다. row는 i값이 0부터 n-1까지 증가하는 값으로 2중 for문이 돌고 있기 때문에 이미 오름차순 정렬이 자동으로 되어 있습니다.

6. "멀티태스킹" 정답코드

```
class Solution {
    public int solution(int[] tasks, long k) {
                int answer = 0;
                int[] sT = new int[tasks.length + 1];
                System.arraycopy(tasks, 0, sT, 1, tasks.length);
                Arrays.sort(sT);
                int rest = tasks.length;
                for(int i = 1; i < sT.length; i++){
                        long time = ((long)rest * (sT[i] - sT[i-1]));
                        if(k < time){
                                long idx = k % rest;
                                int cnt = 0;
                                for(int j = 0; j < tasks.length; j++){
                                        if(tasks[j] >= sT[i]){
                                                if(cnt == idx) return j+1;
                                                cnt++;
                                        }
                                }
                        }
                        else{
                                k -= time;
                                rest--;
                        }
                }
                return -1;
    }
}
```

▶▶ Comment :

```
System.arraycopy(tasks, 0, sT, 1, tasks.length);
```

위 코드는 tasks의 0번 인덱스부터 tasks.length 길이 만큼을 sT의 1번 인덱스부터의 공간에 복사합니다.

7. "최소회의실 개수" 정답코드

```
class Solution {
        public int solution(int[][] meetings){
                ArrayList<int[]> list = new ArrayList<>();
                for(int[] x : meetings){
                    list.add(new int[]{x[0], 1});
                    list.add(new int[]{x[1], 2});
                }
                list.sort((a, b) -> a[0] == b[0] ? b[1] - a[1] : a[0] - b[0]);
                int answer = 0, cnt = 0;
                for(int[] x : list){
                    if(x[1] == 1) cnt++;
                    else cnt--;
                    answer = Math.max(answer, cnt);
                }
                return answer;
        }
}
```

▶▶ Comment :

```
for(int[] x : meetings){
    list.add(new int[]{x[0], 1});
    list.add(new int[]{x[1], 2});
}
list.sort((a, b) -> a[0] == b[0] ? b[1] - a[1] : a[0] - b[0]);
```

회의의 시작시간과 끝나는 시간을 분리해서 리스트에 넣고 시간순으로 정렬을 합니다. 특히 같은 시간에 두 개의 회의가 시작하고, 끝나고를 동시에 한다면 회의가 끝나는 시간이 먼저, 회의가 시작하는 시간이 나중에 오도록 정렬합니다.