```java
package problem1;import java.util.Arrays;
import java.util.Collections;
/* 그리디 문제입니다! 가장 많이 먹는 머리 순으로 순서대로 주면, 최대한 양손을 많이 쓸 수 있습니다. 양손으로 먹이를 주는 것이 끝나면 나머지는 한손으로 주어야 합니다. 한 손으로 주는 것이 최소가 되게 하는것이 핵심!*/
class Solution{public int solution(int[]food){
        int[]sortedFood=Arrays.stream(food).boxed().sorted(Collections.reverseOrder()).mapToInt(i->i).toArray();
        int count = 0; // 급여 횟수
        int currFood = sortedFood[0]; // 현재 급여중인 머리에게 급여해야 할 양
        for (int i=1;i<sortedFood.length;i++){
            int newFood = sortedFood[i]; // 또 다른 머리에게 급여해야 할 양
            if (newFood < currFood) { // 급여 중 다른 머리의 식사가 모두 끝나면
                currFood -= newFood;  // 현재 급여중인 머리에게 해당 횟수만큼 급여 완료
                count += newFood;     // 급여 횟수는 다른 머리의 식사횟수 만큼이 된다.
            } else {  // 현재 급여중인 머리의 급여가 끝나면
                count += currFood; //급여 횟수만큼 추가하고
                currFood = newFood - currFood; // 다른 머리가 이제 '현재 급여중인 머리'가 된다.
            }}count += currFood; // 양손으로 줄 수 있는 급여를 마치고, 한 손 급여만 가능한 케이스
        return count;}}class Test {
    public static void main(String[] args){
        int[] food = {6, 3, 4, 5};System.out.println(new Solution().solution(food));}}
package problem2;/* DP/수학문제입니다. Tabulation 방식으로 계산하면서, 현재까지 구한 값 중 작은 값에
 * *2, *3, *5한 값을 추가해 나갑니다. 값을 추가한 이후에는 2,3,5를 곱할 대상을 더 큰 값으로 바꾸어 줍니다. */
class Solution{public int solution(int n){
    int mult2 = 0, mult3 = 0, mult5 = 0; // 2배, 3배, 5배할 값의 인덱스 초기화
    int[] dp = new int[n];dp[0] = 1;for(int i = 1; i < n; i++) {int valMult2 = 2*dp[mult2];
```

```
25          int valMult3=3*dp[mult3];int valMult5=5*dp[
   mult5];//2/3/5곱한것 중 가장 작은 것을 i번째 숫자로 확정
26          dp[i] = Math.min(valMult2, Math.min(
   valMult3, valMult5));
27          // 아래 조건은 else if로 하면 안됩니다. 왜냐하면 여러
   경우에 동시에 일치할수 있기때문!ex)dp[mult2]=6,dp[mult3]=4,
   dp[mult5]=3인경우,
28          // 2*dp[mult2]와 3*dp[mult3]가 같은 값이므로, 둘
   다 인덱스가 변경되어야 함
29          if(dp[i] == valMult2)  // dp[mult2]의 2배와
   일치하는 값이라면, mult2를 다음 값으로 업데이트 mult2++;
30              if(dp[i] == valMult3)  // dp[mult3]의
   3배와 일치하는 값이라면, mult3를 다음 값으로 업데이트 mult3++;
31                  if(dp[i] == valMult5)  // dp[mult5]
   의 5배와 일치하는 값이라면, mult5를 다음 값으로 업데이트 mult5++;
32                      return dp[n-1];}}class Test{
33      public static void main(String[] args){int n =
   15;System.out.println(new Solution().solution(n
   ));}}
34 package problem3;/* CCW 알고리즘입니다. (Counter-
   ClockWise) 외적을 이용하여 점의 진행 방향을 알아낼 수 있습니다.
35       * 이러한 알고리즘은 단독으로 출제되기 보다는, 더 큰 문제를
   풀고자 할 때 일부분으로 사용되기도 합니다. 그럴 때 이 알고리즘을 알고
   있는지
36       * 아닌지에 따라 구현 속도와 정확도에 차이가 있겠지요! CCW
   알고리즘에 대한 자세한 내용은 원본 문제와 그 답안을 참고해 주세요.
37       * 문제 출처 : https://www.acmicpc.net/problem/
   11758*/
38      class Solution { public String solution(int
   [][] points) {
39          int x1 = points[0][0], y1 = points[0][1];
40          int x2 = points[1][0], y2 = points[1][1];
41          int x3 = points[2][0], y3 = points[2][1];
42          int val = (x1 * y2 + x2 * y3 + x3 * y1) - (
   x2 * y1 + x3 * y2 + x1 * y3);
43          if (val == 0) {return "LINE";} else if (val
    > 0) {return "CCW";} else {return "CW";}}}
44      class Test{public static void main(String[]
   args){
45          int[][] points={{0,0},{0,10},{10,5}};System
   .out.println(new Solution().solution(points));}}
```