题目描述：
为了提升软件编码能力，小王制定了刷题计划，他选了题库中的 n 道题，编号从 0 到 n-1，并计划在 m 天内按照题目编号顺序刷完所有的题目（注意，小王不能用多天完成同一题）。在小王刷题计划中，小王需要用 time[i]的时间完成编号 i 的题目。此外，小王还可以查看答案，可以省去该题的做题时间。为了真正达到刷题效果，小王每天最多直接看一次答案。我们定义 m 天中做题时间最多的一天耗时为 T（直接看答案的题目不计入做题总时间）。请你帮小王求出最小的 T 是多少。
输入描述：
第一行输入为 time，time[i]的时间完成编号 i 的题目
第二行输入为 m，m 表示几天内完成所有题目，1<=m<=180
输出描述：
最小耗时整数 T
补充说明：
示例 1
输入：
999,999,999
4
输出：
0
说明：
在前三天中，小王每天都直接看答案，这样他可以在三天内完成所有的题目并不花任何时间
示例 2
输入：
1,2,2,3,5,4,6,7,8
5
输出：
4
说明：
第一天完成前 3 题，第 3 提看答案；第二天完成第 4 题和第 5 题，第 5 题看答案；第三天完成第 6 和第 7 题，第 7 提看答案；第四天完成第 8 题，直接看答案；第五天完成第 9 题，直接看答案

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] time =
                Arrays.stream(sc.nextLine().split(",")).mapToInt(Integer::parseInt).toArray();
        int day = Integer.parseInt(sc.nextLine());
        System.out.println(getMinT(time, day));
    }

    private static int getMinT(int[] time, int day) {
        if (time.length <= day) {
```

```
            return 0;
        }
        int sum = Arrays.stream(time).sum();
        int T_min = 0;
        int T_max = sum / day + 1;
        while (T_min < T_max) {
            int mid = T_min + (T_max - T_min) / 2;
            if (ok(time, mid,day)) {
                T_min = mid + 1;
            } else {
                T_max = mid;
            }
        }
        return T_max;
    }

    private static boolean ok(int[] time, int mid, int day) {
        int currentT = 0;
        int currentDay = 1;
        int maxT = 0;
        for (int t : time) {
            currentT += t;
            maxT = Math.max(maxT, t);
            if (currentT - maxT > mid) {
                currentT = t;
                currentDay += 1;
                maxT = t;
            }
        }
        return currentDay > day;
    }
}
```