食堂供餐

题目描述：某公司员工食堂以盒饭方式供餐。为将员工取餐排队时间降低为0，食堂的供餐速度必须要足够快。现在需要根据以往员工取餐的统计信息，计算出一个刚好能达成排队时间为0的最低供餐速度。即，食堂在每个单位时间内必须至少做出多少份盒饭才能满足要求。

输入描述：第1行为一个正整数N，表示食堂开餐时长。1 <= N <= 1000。
第2行为一个正整数M，表示开餐前食堂已经准备好的盒饭份数。P1 <= M <= 1000。
第3行为N个正整数，用空格分隔，依次表示开餐时间内按时间顺序每个单位时间进入食堂取餐的人数Pi。1 <= i <= N，0 <= Pi <= 100。

输出描述：一个整数，能满足题目要求的最低供餐速度（每个单位时间需要做出多少份盒饭）。

补充说明：每人只取一份盒饭。
需要满足排队时间为0，必须保证取餐员工到达食堂时，食堂库存盒饭数量不少于本次来取餐的人数。
第一个单位时间来取餐的员工只能取开餐前食堂准备好的盒饭。
每个单位时间里制作的盒饭只能供应给后续单位时间来的取餐的员工。
食堂在每个单位时间里制作的盒饭数量是相同的。

示例1
输入：3
14
10 4 5
输出：3
说明：本样例中，总共有3批员工就餐，每批人数分别为10、4、5。
开餐前食堂库存14份。
食堂每个单位时间至少要做出3份餐饭才能达成排队时间为0的目标。具体情况如下：
第一个单位时间来的10位员工直接从库存取餐。取餐后库存剩余4份盒饭，加上第一个单位时间做出的3份，库存有7份。
第二个单位时间来的4员工从库存的7份中取4份。取餐后库存剩余3份盒饭，加上第二个单位时间做出的3份，库存有6份。
第三个单位时间来的员工从库存的6份中取5份，库存足够。

如果食堂在单位时间只能做出2份餐饭，则情况如下：
第一个单位时间来的10位员工直接从库存取餐。取餐后库存剩余4份盒饭，加上第一个单位时间做出的2份，库存有6份。
第二个单位时间来的4员工从库存的6份中取4份。取餐后库存剩余2份盒饭，加上第二个单位时间做出的2份，库存有4份。
第三个单位时间来的员工需要取5份，但库存只有4份，库存不够。

```
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static int[] employee;
    public static int time;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        time = sc.nextInt();
        int alreadyDishes = sc.nextInt();
        employee = new int[time];
        for (int i = 0; i < time; i++) {
            employee[i] = sc.nextInt();
        }
        int sumOfDishes = 0;
        for (Integer integer : employee) {
            sumOfDishes += integer;
        }
        int minDelivery = 0;
        int maxDelivery = sumOfDishes - alreadyDishes;
        while (minDelivery < maxDelivery) {
            int middle = (maxDelivery + minDelivery) / 2;
            if (getFood(middle, alreadyDishes)) {
```

```
                maxDelivery = middle;
            } else {
                minDelivery = middle + 1;
            }
        }
        System.out.println(minDelivery);
    }

    public static boolean getFood(int middle, int alreadyDishes) {
        boolean result = true;
        for (int i = 0; i < time; i++) {
            alreadyDishes -= employee[i];
            if (alreadyDishes < 0) {
                result = false;
                break;
            }
            alreadyDishes += middle;
        }
        return result;
    }
}
```