题目描述：
一个工厂有 m 条流水线，来并行完成 n 个独立的作业，该工厂设置了一个调度系统，在安排作业时，总是优先执行处理时间最短的作业。
现给定流水线个数 m，需要完成的作业数 n，每个作业的处理时间分别为 t1,t2…tn。请你编程计算处理完所有作业的耗时为多少？
当 n>m 时，首先处理时间短的 m 个作业进入流水线，其他的等待，当某个作业完成时，依次从剩余作业中取处理时间最短的进入处理。

输入描述：
第一行为 2 个整数（采用空格分隔），分别表示流水线个数 m 和作业数 n；
第二行输入 n 个整数（采用空格分隔），表示每个作业的处理时长 t1,t2…tn。
0< m,n<100，0<t1,t2…tn<100。
注：保证输入都是合法的。
输出描述：
输出处理完所有作业的总时长
补充说明：

示例 1
输入：
3 5
8 4 3 2 10
输出：
13
说明：
1、先安排时间为 2、3、4 的 3 个作业。
2、第一条流水线先完成作业，然后调度剩余时间最短的作业 8。
3、第二条流水线完成作业，然后调度剩余时间最短的作业 10。
4、总工耗时就是第二条流水线完成作业的时间 13（3+10）。

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String str = scanner.nextLine();
        String[] arr = str.split(" ");
        int pipelineCount = Integer.parseInt(arr[0]);
        int taskCount = Integer.parseInt(arr[1]);

        int[] costArr = new int[taskCount];
        for (int    i = 0; i < taskCount; i++) {
            costArr[i] = scanner.nextInt();
        }

        Arrays.sort(costArr);
```

```
            if (pipelineCount >= taskCount) {
                System.out.println(costArr[taskCount - 1]);
                return;
            }

            int index = 0;
            int[] pipelineCost = new int[pipelineCount];
            for (int i = 0; i < taskCount; i++) {
                pipelineCost[index] += costArr[i];
                index++;
                if (index == pipelineCount) {
                    index = 0;
                }
            }
            Arrays.sort(pipelineCost);
            System.out.println(pipelineCost[pipelineCount - 1]);

        }
    }
```