AcWing-3321：ATM队列

N 个人（编号 $1 \sim N$），排成一队在 ATM 机前准备取钱。

初始时，队列按编号升序的顺序排列。

第
i 个人需要取
A_i 元钱。

一个人一次最多可以取 X 元钱。

当轮到某个人取钱时，如果其需要的钱的数量大于
X，则只能先取
X 元钱，然后去队尾重新排队，等待下次轮到他取钱时，继续去取。

当一个人取够钱时，他就会拿着钱离开队列。

现在，请你确定所有人离开队列的顺序。

输入格式

第一行包含整数
T，表示共有
T 组测试数据。

每组数据第一行包含两个整数
N 和
X。

第二行包含
N 个整数
A_i。

时/空限制：	2s / 64MB
总通过数：	989
总尝试数：	2470
来源：	Google Kickstart2020 Round F Problem A
算法标签	

输出格式

每组数据输出一个结果，每个结果占一行。

结果表示为 `Case #x: y`，其中 x 为组别编号（从 1 开始），y 是空格隔开的 N 个整数，表示所有人离开队列的顺序列表。

数据范围

$$\begin{aligned} &60\%\text{数据满足，}1 \le T \le 100\text{，}1 \le N \le 100, \\ &\text{另外}40\%\text{数据满足，}1 \le T \le 10\text{，}1 \le N \le 10^5, \\ &100\%\text{数据满足，}1 \le Ai \le 10^9\text{，}1 \le X \le 10^9\text{。} \end{aligned} \tag{1}$$

输入样例：

```
2
3 3
2 7 4
5 6
9 10 4 7 2
```

输出样例：

```
Case #1: 1 3 2
Case #2: 3 5 1 2 4
```

样例解释

对于测试数据 1，共 3 个人，单轮最大取钱数量为 3。

取钱过程如下：

1. 初始时，队列为 [1,2,3]，第一个人需要 2 元钱，取完离队。
2. 此时，队列为 [2,3]，第二个人需要 7 元钱，所以先取 3 元，然后去队尾继续排队。
3. 此时，队列为 [3,2]，第三个人需要 4 元钱，所以先取 3 元，然后去队尾继续排队。
4. 此时，队列为 [2,3]，第二个人需要 4 元钱，所以先取 3 元，然后去队尾继续排队。
5. 此时，队列为 [3,2]，第三个人需要 1 元钱，取完离队。
6. 此时，队列为 [2]，第二个人需要 1 元钱，取完离队。
7. 队列为空。

离队顺序为 [1,3,2]。

对于测试数据 2，共 5 个人，单轮最大取钱数量为 6。

取钱过程如下：

1. 初始时，队列为 [1,2,3,4,5]，第一个人需要 9 元钱，所以先取 6 元，然后去队尾继续排队。
2. 此时，队列为 [2,3,4,5,1]，第二个人需要 10 元钱，所以先取 6 元，然后去队尾继续排队。
3. 此时，队列为 [3,4,5,1,2]，第三个人需要 4 元钱，取完离队。
4. 此时，队列为 [4,5,1,2]，第四个人需要 7 元钱，所以先取 6 元，然后去队尾继续排队。
5. 此时，队列为 [5,1,2,4]，第五个人需要 2 元钱，取完离队。
6. 此时，队列为 [1,2,4]，这三个人还需取得的钱数均不超过 6 元，所以可以顺次取完离队。

离队顺序为 [3,5,1,2,4]。

本题就是个简单的排序问题，有两种解决思路：

- 第一种是使用队列模拟，不断出队入队从而模拟出最终的排序。
- 第二种是进行给定规则的排序——如果取钱次数相同，就按照序号排序；不相同，按照取钱次数排序。即可自定义`cmp`排序函数进行`sort`排序即可。

Code：

```
//
//  main.cpp
//  3321-ATM队列
//
```

```cpp
//  Created by MacBook Pro on 2023/8/6.
//

#include <iostream>
#include <algorithm>
#include <cstring>
#include <stdio.h>
#include <cmath>
using namespace std;

struct person{
    int num;    //编号
    int times;  //取钱次数
};

vector<person> que;

bool cmp(const person& p1,const person& p2){
    if(p1.times!=p2.times) return p1.times<p2.times;
    else{
        return p1.num<p2.num;
    }
}

int main() {
    int T;
    scanf("%d",&T);
    for(int cases=1;cases<=T;cases++){
        que.clear();
        int N,X;
        scanf("%d%d",&N,&X);
        for(int i=1;i<=N;i++){
            int tmp;
            scanf("%d",&tmp);
            que.push_back({i,int(ceil(double(tmp)/X))});
        }
        sort(que.begin(),que.end(),cmp);
        printf("Case #%d:",cases);
        for(int i=0;i<que.size();i++){
            printf(" %d",que[i].num);
        }
        printf("\n");
    }
    return 0;
}
```

需要额外注意的是取钱次数的计算，不能简单地用 需要的钱/单次最多取出数额 进行计算。本人采用：

$$int(ceil(double(\text{需要的钱})/X)) \tag{2}$$

代码提交状态：Accepted