## Problem 1. Cow College

Farmer John 计划为奶牛们新开办一所大学！

有 NN（1≤N≤1051≤N≤105）头奶牛可能会入学。每头奶牛最多愿意支付 cici 的学费（1≤ci≤1061≤ci≤106）。 Farmer John 可以设定所有奶牛入学需要支付的学费。如果这笔学费大于一头奶牛愿意支付的最高金额，那么这头奶牛就不会入学。Farmer John 想赚尽可能多的钱，从而可以给他的讲师提供一笔可观的工资。请求出他能赚到的钱的数量，以及此时应当收取多少学费。

**输入格式（从终端 / 标准输入读入）：**

输入的第一行包含 NN。第二行包含 NN 个整数 c1,c2,…,cNc1,c2,…,cN，其中 cici 是奶牛 ii 愿意支付的最高学费金额。

**输出格式（输出至终端 / 标准输出）：**

输出 Farmer John 可以赚到的最大金额以及最优情况下他应该收取的学费。如果有多个解，输出收取学费最小的解。

**注意这个问题涉及到的整数可能需要使用 64 位整数型（例如，Java 中的 "long"，C/C++ 中的 "long long"）。**

**输入样例：**

4

1 6 4 6

**输出样例：**

12 4

如果 Farmer John 收费 44，那么 33 头奶牛将会入学，从而使他赚取 3⋅4=123⋅4=12 的金额。

**测试点性质：**

* 测试点 2-4 满足 ci≤1,000ci≤1,000。
* 测试点 5-8 满足 N≤5,000N≤5,000。
* 测试点 9-12 没有额外限制。

供题：Freddie Tang

## Problem 2. Feeding the Cows

Farmer John 有 NN（1≤N≤1051≤N≤105）头奶牛，每头奶牛的品种是更赛牛（Guernsey）或荷斯坦牛（Holstein）之一。她们沿水平方向排成一行，奶牛们占据的位置编号为 1…N1…N。

由于奶牛们都饿了，FJ 决定在 1…N1…N 中的某些位置上种植草地。更赛牛和荷斯坦牛喜欢不同类型的草，所以如果 Farmer John 决定在某个位置种草，他必须选择种植更赛牛喜欢的草或荷斯坦牛喜欢的草——他不能在同一个位置同时种两种草。种植的每一片草地都可以喂饱数量不限的相应品种的奶牛。

每头奶牛愿意移动至多 KK（0≤K≤N−10≤K≤N−1）个位置以前往一个草地。求出喂饱所有奶牛所需种植的最小草地数量。此外，输出一种使用最小草地数量喂饱所有奶牛的种植方案。任何满足上述条件的方案均视为正确。

**输入格式（从终端 / 标准输入读入）：**

每个测试用例包含 TT 个子测试用例，为一种奶牛的排列。输入的第一行包含 TT（1≤T≤101≤T≤10）。以下是 TT 个子测试用例。

每个子测试用例的第一行包含 NN 和 KK。第二行包含一个长度为 NN 的字符串，其中第 ii 个字符表示第 ii 头奶牛的品种（G 表示更赛牛，H 表示荷斯坦牛）。

**输出格式（输出至终端 / 标准输出）：**

对于 TT 个子测试用例中的每一个，输出两行。第一行输出喂饱所有奶牛所需的最小草地数量。第二行输出一个长度为 NN 的字符串，表示一种使用最小草地数量喂饱所有奶牛的种植方案。第 ii 个字符表示第 ii 个位置，若不种草则为 '.'，若种植喂饱更赛牛的草则为 'G'，若种植喂饱荷斯坦牛的草则为 'H'。任何合法的方案均可通过。

**输入样例：**

6

5 0

GHHGG

5 1

GHHGG

5 2

GHHGG

5 3

GHHGG

5 4

GHHGG

2 1

GH

**输出样例：**

5

GHHGG

3

.GH.G

2

..GH.

2

...GH

2

...HG

2

HG

注意对于某些子测试用例，存在多种可通过的方案使用最小数量的草地。例如，在第四个子测试用例中，以下是另一个可以通过的答案：

.GH..

这个方案在第二个位置种植一块喂饱更赛牛的草地以及在第三个位置种植一块喂饱荷斯坦牛的草地。这使用了最小数量的草地并确保了所有奶牛都在她们喜欢的草地的 3 个位置以内。

**测试点性质：**

* 测试点 2-4 满足 N≤10N≤10。
* 测试点 5-8 满足 N≤40N≤40。
* 测试点 9-12 满足 N≤105N≤105。

供题：Mythreya Dharani

## Problem 3. Reverse Engineering

Elsie 有一个程序，接受一个 NN（1≤N≤1001≤N≤100）个变量的数组 b[0],…,b[N−1]b[0],…,b[N−1] 作为输入，其中每个变量等于 0 或 1，并且返回对输入数组应用一系列 if / else if / else 语句的结果。每个语句检查至多一个输入变量的值，并返回 0 或 1。这类程序的一个例子是：

if (b[1] == 1) return 1;

else if (b[0] == 0) return 0;

else return 1;

例如，如果上方程序的输入是 "10"（即 b[0]=1b[0]=1 及 b[1]=0b[1]=0），那么输出应当为 1。

Elsie 告诉了 Bessie 对于 MM（1≤M≤1001≤M≤100）个不同输入的正确输出。Bessie 现在正试图对 Elsie 的程序进行逆向工程。不幸的是，Elsie 可能说了谎；可能不存在上述形式的程序行为与 Elsie 所说的均一致。

对于 TT（1≤T≤101≤T≤10）个子测试用例中的每一个，判断 Elsie 是否一定在说谎。

**输入格式（从终端 / 标准输入读入）：**

输入的第一行包含 TT，为子测试用例的数量。

每一个子测试用例的第一行包含两个整数 NN 和 MM，以下 MM 行，每行包含一个由 NN 个 0 或 1 组成的字符串，表示一个输入（即 b[0]…b[N−1]b[0]…b[N−1] 的值），以及另一个字符（0 或 1）表示输出。相邻的子测试用例之间用空行分隔。

**输出格式（输出至终端 / 标准输出）：**

对于每一个子测试用例，输出一行，包含 "OK" 或 "LIE"，分别表示 Elsie 可能没有说谎或是一定在说谎。

**输入样例：**

4

1 3

0 0

0 0

1 1

2 4

00 0

01 1

10 1

11 1

1 2

0 1

0 0

2 4

00 0

01 1

10 1

11 0

**输出样例：**

OK

OK

LIE

LIE

以下是第一个子测试用例的一个合法的程序：

if (b[0] == 0) return 0;

else return 1;

以下是第一个子测试用例的另一个合法的程序：

if (b[0] == 1) return 1;

else return 0;

以下是第二个子测试用例的一个合法的程序：

if (b[1] == 1) return 1;

else if (b[0] == 0) return 0;

else return 1;

显然，对于第三个子测试用例不存在对应的合法的程序，因为 Elsie 的程序一定始终对相同的输入产生相同的输出。

可以证明对于最后一个子测试用例不存在对应的合法的程序。

**测试点性质：**

* 测试点 2-3 满足 N=2N=2。
* 测试点 4-5 满足 M=2M=2。
* 测试点 6-12 没有额外限制。

供题：Benjamin Qi