布丁（pudding）

本题为交互题。

【题目描述】

小 L 和小 S 都非常喜欢香甜而软糯的布丁。在品尝过各种布丁后，她们将所有布丁的美味度量化为不超过 4500 的正整数。

小 L 正在学习制作布丁。因熟练程度有限，她只能制作出美味度不超过常数 m 的布丁。在某次尝试中，小 L 成功制作出了一块美味度为 w 的布丁。小 S 想品尝小 L 制作的布丁，但需要按照小 L 要求的方式求出她制作的布丁的美味度。

具体地，小 S 可以从商店购买若干块布丁并交给小 L。小 L 会把自己制作的布丁混入其中，并将这些布丁按照美味度升序排序。然后，小 L 会计算全部相邻布丁美味度的最大公约数之和并告诉小 S。最后，她会将所有小 S 买来的布丁吃掉。形式化地，设小 S 购买了 k 块布丁，美味度分别为 $a_0, a_1, \ldots, a_{k-1}$。将 $[a_0, a_1, \ldots, a_{k-1}, w]$ 按升序排序后的结果记为 $[b_0, b_1, \ldots, b_{k-1}, b_k]$，则小 L 会将 $\sum_{i=1}^{k} \gcd(b_{i-1}, b_i)$ 的值告诉小 S。

由于前往商店的时间成本与购买布丁的经济成本都很高，小 S 希望尽可能减少购买的次数以及购买的布丁总数。你需要帮助小 S 制定购买策略，以求出小 L 制作的布丁的美味度。

【实现细节】

选手不需要，也不应实现 `main` 函数。

选手需要确保提交的程序包含头文件 `pudding.h`，即在程序开头加入以下代码：

```
#include "pudding.h"
```

选手需要在提交的程序源文件 `pudding.cpp` 中实现以下两个函数：

```
void init(int c, int t);
```

- c, t 分别表示测试点编号与测试数据组数。$c = 0$ 表示该测试点为样例。
- 对于每个测试点，该函数会在程序开始运行时被交互库调用恰好一次。

```
int find_tastiness(int c, int m);
```

- c, m 分别表示测试点编号与小 L 制作的布丁的美味度的上界；
- 该函数需要返回一个正整数 w，表示小 L 制作的布丁的美味度。
- 对于每个测试点，该函数会被交互库调用恰好 t 次。

选手可以通过调用以下函数进行一次询问：

```
int query_tastiness(std::vector<int> a);
```

- a 表示小 S 购买的布丁的美味度的序列。选手需要确保 a 非空，且其中的每个元素均为不超过 4500 的正整数。
- 该函数会返回小 L 告诉小 S 的值，具体含义如【题目描述】中所示。
- 选手需要确保交互库每次调用 `find_tastiness` 时，调用该函数的次数不超过 15，且调用该函数时传入的 a 的长度之和不超过 3000。

在任何情况下，交互库运行所需时间均不会超过 1.5 秒，所用内存不会超过 64 MiB。

本试题目录下的 `template_pudding.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

【测试程序方式】

选手可以在本题目录下使用如下命令编译得到可执行文件：

```
g++ grader.cpp pudding.cpp -o pudding -O2 -std=c++14 -static
```

对于编译得到的可执行文件 `pudding`：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行包含三个非负整数 c, t, m。
  - 第二行包含 t 个正整数，分别表示每组测试数据中 w 的值。
- 可执行文件将输出以下格式的数据至标准输出：
  - 若 t 次调用 `find_tastiness` 的返回值均正确，则：
    - 输出的第一行为 `Correct!`；
    - 输出的第二行为 `Max queries used: Q`，其中 Q 表示所有测试数据中调用 `query_tastiness` 的次数的最大值；
    - 输出的第三行为 `Max total puddings queried: S`，其中 S 表示所有测试数据中调用 `query_tastiness` 时传入的 a 的长度之和的最大值。
  - 若至少一次调用 `find_tastiness` 的返回值不正确，则只会输出一行 `Wrong answer.`。
- 若调用 `query_tastiness` 时传入的参数不符合要求，或调用次数超过上限，则可执行文件会向**标准错误流**输出错误信息，并返回 -1。
- 选手可以在运行可执行文件时启用 `-v` 或 `--verbose` 参数，此时可执行文件将会额外输出以下内容：
  - 每次调用 `find_tastiness` 的返回值、正确性、调用 `query_tastiness` 的次数与传入的 a 的长度之和。
  - 每次调用 `query_tastiness` 时传入的参数、计算过程以及返回值。
  - 程序最终获得的分数比例，具体可见【评分方式】一节。

【样例 1 输入】

```
0 2 197
26 121
```

【样例 1 输出】

```
Correct!
Max queries used: 2
Max total puddings queried: 5
```

【样例 1 解释】

对于第一组测试数据，小 L 制作的布丁的美味度为 26。以下是一种可能的交互过程：

- 调用 `query_tastiness([2026, 7, 20])`，则 $b = [7, 20, 26, 2026]$，因此函数返回 $\gcd(7, 20) + \gcd(20, 26) + \gcd(26, 2026) = 1 + 2 + 2 = 5$；
- 调用 `query_tastiness([13, 52])`，则 $b = [13, 26, 52]$，因此函数返回 $\gcd(13, 26) + \gcd(26, 52) = 13 + 26 = 39$；
- 返回 26，答案正确；
- 调用 `query_tastiness` 的次数为 2，调用 `query_tastiness` 时传入的 a 的长度之和为 $3 + 2 = 5$。

【样例 2】

见选手目录下的 ***pudding/pudding2.in*** 与 ***pudding/pudding2.ans***。
该样例满足测试点 1 的约束条件。

【样例 3】

见选手目录下的 ***pudding/pudding3.in*** 与 ***pudding/pudding3.ans***。
该样例满足测试点 2 的约束条件。

【样例 4】

见选手目录下的 ***pudding/pudding4.in*** 与 ***pudding/pudding4.ans***。
该样例满足测试点 3 的约束条件。

【数据范围】

对于所有测试数据，均有：

- $1 \le t \le 3000$；
- $1 \le m \le 3000$，$1 \le w \le m$。

测试点编号	分值	$t=$	$m=$	特殊性质
1	10	35	35	无
2	20	430	3,000	A
3	70	3,000		无

特殊性质 A：w 为质数。

【评分方式】

注意：

- 选手不应当通过非法方式获取交互库的内部信息，如试图直接读取 w 的值，或直接与标准输入、输出流进行交互。此类行为将被视为作弊。
- 交互库不是适应性的，即每次调用 `find_tastiness` 时 w 的值就已经确定，不会随交互过程变化。
- 最终的评测交互库与样例交互库的实现不同。

若 `find_tastiness` 函数的返回值不正确，或调用 `query_tastiness` 时传入的参数不符合要求，则相应测试点得 0 分。

在上述条件基础上：

- 对于每个测试点，设 Q 表示所有测试数据中调用 `query_tastiness` 的次数的最大值，S 表示所有测试数据中调用 `query_tastiness` 时传入的 a 的长度之和的最大值，$score$ 表示该测试点的分值，则程序获得 $\lfloor f(Q) \cdot g(S) \cdot score \rfloor$ 分，其中 f 与 g 的计算方式如下：

Q	$f(Q)$
$Q \le 4$	1
$5 \le Q \le 15$	0.7^{Q-4}

S	$g(S)$
$S \le 35$	1
$36 \le S \le 75$	$1-(S-35)/100$
$76 \le S \le 235$	$0.2+\sqrt{(235-S)/1000}$
$236 \le S \le 3000$	$0.2 \times 2^{-(S-235)/1500}$