蛋糕（cake）

这是一道交互题。

【题目描述】

Lavi 是一位很喜欢吃蛋糕的小星使。她认为每块蛋糕的美味度都可以定义为一个正整数。

有一天，她的好朋友 Sally 买来了一块蛋糕。这块蛋糕的美味度是一个不大于 W 的正整数 d，但 Lavi 此时只知道 W 的值。现在 Lavi 想要求出 d 的值。

运用星使的力量，Lavi 可以制作出至多 N 块美味度分别不超过 $W+200$ 的蛋糕并告知 Sally；然后 Sally 会偷偷地将买来的那块蛋糕混进这些蛋糕中，最后 Sally 会将这些蛋糕**按美味度升序排序**。由于这些不同美味度的蛋糕在外表上没有任何区别，所以 Lavi 暂时不能分辨出哪块蛋糕是 Sally 买来的。不过 Sally 有很强的记忆力，因此她清晰地记得排序后每块蛋糕的美味度。

将蛋糕排序后，Lavi 可以多次向 Sally 进行以下询问：

- Lavi 选出若干块蛋糕，将它们分为不相交的两组，然后 Sally 会告知 Lavi 两组蛋糕美味度之和的大小关系。

  形式化地，假设 Lavi 制作了 m 块蛋糕，记 $a_0, a_1, a_2, \ldots, a_m$ 为将 Sally 买来的蛋糕混入并排序后每块蛋糕的美味度，那么 Lavi 每次需要给出两个非空下标集合 S_1, S_2，满足 $S_1, S_2 \subseteq \{0, 1, 2, \ldots, m\}$ 且 $S_1 \cap S_2 = \varnothing$。Sally 会告知 Lavi $\sum_{i \in S_1} a_i$ 与 $\sum_{i \in S_2} a_i$ 的大小关系。

现在 Lavi 需要你的帮助！但她提醒你，询问次数过多会对 Sally 的记忆力提出很大的考验，因此 Sally 对询问次数做出了一定限制。具体地，Sally 会在 Lavi 制作蛋糕前给出一个正整数 K，如果 Lavi 作出了多于 K 次询问，那么你获得的分数会随询问次数增加而减少。

请协助 Lavi 决定制作蛋糕的策略以及随后的询问策略。

【实现细节】

选手不需要，也不应该实现 `main` 函数。

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

```
#include "cake.h"
```

选手需要在提交的程序源文件 `cake.cpp` 中实现以下两个函数：

```
std::vector<int> bake_cakes(int N, int W, int K);
```

- N 表示 Lavi 最多可制作的蛋糕数。
- W 表示 Sally 买来的蛋糕的美味度上界。
- K 表示询问次数阈值。

- 该函数需要返回一个正整数数组 c，表示 Lavi 制作出每块蛋糕的美味度，其中：
  - c 的长度 m 不能超过 N；
  - 对于所有 $0 \le i \le m-1$，均有 $1 \le c_i \le W + 200$。
- 对于每个测试点，该函数会被交互库调用恰好一次，且在所有 `find_tastiness` 函数调用之前。

```
int find_tastiness(int m, int W, int K);
```

- m 表示 Lavi 制作出的蛋糕数，也就是说，包含 Sally 买来的蛋糕后实际蛋糕数为 $m+1$。
- W 表示 Sally 买来的蛋糕的美味度上界。
- K 表示询问次数阈值。
- 该函数需要返回一个正整数 d，表示 Lavi 求出的 Sally 买来的蛋糕的美味度。
- 对于每个测试点，该函数可能会被交互库调用多次，每次调用之间是独立的。

在该函数中，你可以调用以下函数：

```
int compare_tastiness(std::vector<int> S1, std::vector<int>
  S2);
```

- S_1, S_2 分别为两组蛋糕的下标集合，你需要满足 S_1, S_2 非空且其中的元素互不相同，即 $S_1, S_2 \subseteq \{0, 1, 2, \ldots, m\}$ 且 $S_1 \cap S_2 = \varnothing$。
- 该函数会返回一个整数 $r \in \{-1, 0, 1\}$ 代表 $v_1 = \sum_{i \in S_1} a_i$ 与 $v_2 = \sum_{i \in S_2} a_i$ 的大小关系，其中 $r = -1$ 代表 $v_1 < v_2$，$r = 0$ 代表 $v_1 = v_2$，$r = 1$ 代表 $v_1 > v_2$。
- 在一次 `find_tastiness` 的调用中，你可以调用该函数至多 100 次。

评测程序不是适应性的，Sally 买来的蛋糕的美味度 d 在每次 `find_tastiness` 的调用前就已经确定。

注意：在任何情况下，交互库运行所需时间均不会超过 6 秒。

【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程序：

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

对于编译得到的可执行程序：

- 可执行文件将从标准输入读入以下格式的数据：
  - 输入的第一行包含四个正整数 N, W, K, T。
  - 输入的第二行包含 T 个正整数 $d_1, d_2, \ldots, d_T$，分别表示每次调用 `find_tastiness` 时 Sally 买来的蛋糕的美味度。
- 可以通过在运行时启用 `-v` 或 `--verbose` 参数来输出更详细的交互流程。在未启用该参数的情况下，程序会在每次调用完 `find_tastiness` 后输出返回

值的正确性以及 `compare_tastiness` 的调用次数，并在所有调用完成后输出 `compare_tastiness` 的最大调用次数。启用 `-v` 或 `--verbose` 参数后，程序将会额外输出以下内容：

- 调用 `bake_cakes` 后函数的返回值。
- 每次 `compare_tastiness` 被调用时传递的参数，比较信息以及比较结果。

【样例 1 输入】

```
40 20 5 3
19 7 20
```

【样例 1 输出】

```
Correct: found 19 in 4 compares
Correct: found 7 in 5 compares
Correct: found 20 in 5 compares
Correct. Max compare count is 5
```

【样例 1 解释】

交互库将进行以下调用：

```
bake_cakes(40, 20, 5);
```

Lavi 可以制作至多 40 个蛋糕，Sally 买来的蛋糕的美味度上界为 20，询问次数阈值为 5。

一种可能的返回数组为 $[12, 1, 22, 9, 19, 1, 12, 12, 25]$。

接下来交互库将进行三次以下调用：

```
find_tastiness(9, 20, 5);
```

其中每次调用时 Sally 买来的蛋糕的美味度分别为 $19, 7, 20$。

- Sally 买来的蛋糕的美味度为 19 时，所有蛋糕排序后美味度分别为 $[1, 1, 9, 12, 12, 12, 19, 19, 22, 25]$。此时，
  - 若调用 `compare_tastiness([0, 2, 4], [1, 3, 5])`，则 S_1 中蛋糕美味度之和为 $1 + 9 + 12 = 22$，S_2 中蛋糕美味度之和为 $1 + 12 + 12 = 25$，因此该函数会返回 -1；
  - 若调用 `compare_tastiness([8, 2, 6], [5, 0, 9])`，则 S_1 中蛋糕美味度之和为 $22 + 9 + 19 = 50$，S_2 中蛋糕美味度之和为 $12 + 1 + 25 = 38$，因此该函数会返回 1；

  - 若调用 `compare_tastiness([0, 4, 7], [1, 3, 6])`，则 S_1 中蛋糕美味度之和为 $1+12+19=32$，S_2 中蛋糕美味度之和为 $1+12+19=32$，因此该函数会返回 0。

- Sally 买来的蛋糕的美味度为 7 时，所有蛋糕排序后美味度分别为 $[1,1,7,9,12,12,12,19,22,25]$。此时，
  - 若调用 `compare_tastiness([0, 1, 3], [6])`，则 S_1 中蛋糕美味度之和为 $1+1+9=11$，S_2 中蛋糕美味度之和为 19，因此该函数会返回 -1。

【数据范围】

对于所有测试数据，均有：

- $1 \le N \le 3 \times 10^3$，$1 \le W \le 10^9$，$1 \le K \le 100$，$1 \le T \le 2 \times 10^3$；
- 对于所有 $1 \le i \le T$，均有 $1 \le d_i \le W$。

测试点编号	分值	$N=$	$W=$	$K=$	$T \le$
1	7	$3,000$	10^2	10^2	10^2
2	8	3	3	1	3
3	30	40	10^9	30	$2,000$
4	55	$3,000$	$2,000$	7	

【评分方式】

在任意测试用例中，如果 `bake_cakes` 的返回值不符合实现细节中的约束条件，或者 `compare_tastiness` 的调用不符合实现细节中的约束条件，或者 `find_tastiness` 的返回值不正确，则该子任务得 0 分。

对于每个测试点，记 Q 为该测试点所有 `find_tastiness` 的调用中，`compare_tastiness` 调用次数的最大值，则程序得到的分数将按照以下方式计算：

- 在测试点 $1,2$ 中，若 $Q \le K$，则得分为该测试点的分值，否则得分为 0；
- 在测试点 3 中，得分为 $\max(30-3\cdot\max(Q-K,0),0)$；
- 在测试点 4 中，得分为 $\max(55-11\cdot\max(Q-K,0),0)$。