第一題：聲納探測系統 (sonnakoto)

本題為互動題。

問題敘述

踢歐埃王國是一個充滿了神奇魔法的國度，其東岸是整條帶狀的懸崖，緊鄰著一片邊不著際的海洋。最近有人從靠近岸邊的海底發現了不規則形狀的大塊金屬碎片。據推測，很有可能是某種被稱為『太平洋浮標』的大型海上設施被炸掉之後，被洋流帶到岸邊深處的碎片。

為了盡快打撈這些大塊金屬碎片並還原事情經過，你決定先派出許多小型快艇，利用聲納的方式將這些金屬碎片定位，縮小到一公尺左右的範圍。為此，你事先將海岸線由北到南劃分成了 10^9 個區間，每個區間長度都是一公尺，編號為 $1, 2, \ldots, 10^9$。超過東岸的海域範圍也可以按這個順序編號，所以紀錄上允許負數、或是超過 10^9 的區間編號。

根據漁民回報，有 n 個大型金屬碎片散落在區間編號介於 L 與 R 之間的東岸海域，每個區間編號至多僅含一個大型金屬碎片；而且不會有任何金屬碎片散落在編號小於 L、或是大於 R 的區間。你總共有 m 艘快艇。每隔一個小時，你可以派遣這些快艇到指定的區域進行聲納探測。指令是這樣的，你可以決定每一艘快艇 i 的起始掃描區間編號 p_i 以及欲掃描的總長度 k。在這一個小時內，這艘快艇會掃描過編號為 $p_i, p_i + 1, \ldots, p_i + k - 1$ 這些區間 (稱之為一「區間範圍」)，並且得知該區間範圍內『存在』或『不存在』大塊金屬碎片。

不過，這些快艇上面安裝的聲納探測系統有點小毛病：m 艘快艇必須同時**全部**啟動。而且，每一次啟動時，每一艘快艇所掃描的區間範圍之總長度必須是相同的。此外，這些區間也不能重疊：一但掃描的區間重疊，可能會因為訊息干擾而導致**整個**系統崩潰。

請寫一個程式即時地啟動聲納探測系統，在儘量短的時間內找到 n 個大型金屬碎片所在的區間。

實作細節

你需要實作以下函式：

```
std::vector<int> detect_debris(int n, int m, int L, int R);
```

- 對於每一筆測試資料，正式評分程式會呼叫你實作的 `detect_debris()` 函式 $T \leq 5$ 次。
- n 代表金屬碎片所在的區間數量。
- m 代表快艇的總數。

- L 與 R 代表這些金屬碎片所在的區間範圍。
- `detect_debris()` 回傳一個長度恰好為 n 的陣列，代表偵測到的所有金屬碎片位置，任意順序排列皆可。

你的程式可以呼叫以下的函式：

```
std::vector<int> activate_sonar_system(int k, const std::vector<int>& p);
```

- 對於每一次 `detect_debris()` 呼叫，你的函式可以呼叫 `activate_sonar_system()` 至多 10 000 次。
- k 為每一艘快艇欲進行掃描的區間長度，它必須滿足 $1 \le k \le 10^9$。
- $p[0], p[1], \ldots, p[m-1]$ 為一個長度為 m 的序列，代表每一艘快艇被指派到的起始掃描位置。對於所有 i，傳入之 $p[i]$ 必須滿足 $-2 \times 10^9 \le p[i] \le 2 \times 10^9$。
- `activate_sonar_system()` 會回傳一個長度為 m 的序列

$$response[0], response[1], \ldots, response[m-1].$$

對於所有的 i，$response[i]$ 的值可以是 0 或 1。當該值為 1 時，代表著區間範圍 $[p[i], p[i]+k-1]$ 內有掃描到金屬碎片；反之則不存在金屬碎片。

互動範例

一個可能被評為 `Accepted` 的互動例子顯示如下：

評分程式端	參賽者端
呼叫 `detect_debris(` $2, 2, 1, 8$ `)`。	
	呼叫 `activate_sonar_system(` $4, [1, 5]$ `)`。
回傳 $[1, 0]$。	
	呼叫 `activate_sonar_system(` $2, [1, 3]$ `)`。
回傳 $[0, 1]$。	
	回傳 $[3, 4]$。

測資限制

- $1 \le T \le 5$。
- $1 \le n \le 10\,000$。
- $1 \le m \le 100$。
- $0 \le L \le R \le 10^9$。
- 所有金屬碎片的位置在呼叫 `detect_debris()` 前就已經固定了。

評分說明

對於每一筆測試資料，你將會得到一個**分數權重** P。該分數權重與**詢問次數上界** Q_{limit} 與**滿分詢問次數** Q_{full} 相關。若在任何一次 `detect_debris()` 呼叫中，你的程式回傳了錯誤的區間編號，那麼 $P = 0$。否則，令 Q 為一次 `detect_debris()` 呼叫中，你的函式呼叫 `activate_sonar_system()` 的次數。此時分數權重的定義如下：

$$P = \begin{cases} 0.0 & \text{若 } Q > Q_{\text{limit}}\text{；} \\ 1 - \dfrac{Q - Q_{\text{full}}}{Q_{\text{limit}} - Q_{\text{full}}} & \text{若 } Q_{\text{full}} < Q \le Q_{\text{limit}}\text{；} \\ 1.0 & \text{若 } Q \le Q_{\text{full}}\text{。} \end{cases}$$

本題共有 5 組子任務，條件限制如下所示。每一組可有一或多筆測試資料，你在該子任務的得分為所有組內測試資料的 P 值的最小值，乘以該子任務的總分。

子任務	分數	輸入限制	Q_{full}	Q_{limit}
1	2	$n = 1, m = 1, L = 1, R = 10\,000$	14	15
2	10	$n \le 100, m = 1, L = 1, R = 10\,000$	1 111	1 888
3	3	$n = 1, m = 99, L = 1, R = 10\,000$	2	4
4	4	$n \le 10\,000, m = 100, L = 1, R = 10\,000$	100	200
5	81	$n \le 10\,000, m = 100, L = 1, R = 10^9$	2 888	8 888

範例評分程式

範例評分程式採用以下格式輸入：

$$
\begin{array}{l}
T \\
n_1 \ \ m_1 \ \ L_1 \ \ R_1 \\
a_{1,1} \ \ a_{1,2} \ \ \cdots \ \ a_{1,n} \\
\vdots \\
n_T \ \ m_T \ \ L_T \ \ R_T \\
a_{T,1} \ \ a_{T,2} \ \ \cdots \ \ a_{T,n_T}
\end{array}
$$

第一列的 T 代表呼叫 `detect_debris()` 的次數。對於第 t 次的呼叫 ($1 \le t \le T$)，範例評分程式會傳入 n_t, m_t, L_t, R_t。而實際的金屬碎片所在的區間編號為 $a_{t,1}, a_{t,2}, \ldots, a_{t,n_t}$。在本地測試時，範例評分程式會將 `detect_debris()` 回傳的結果依次輸出。

在本地自行進行 stdin/stdout 測試時，請將 `stub.cpp` 檔案內的**第 43 至 44 行註解**與**第 41 至 42 行註解**交換。將之與您所撰寫的檔案一同編譯後，便能在終端機上手動測試。請參考 `1A_sample.cpp` 檔案。

請注意: 使用自己上傳的測試資料進行測試時，沒有下列 MSG 描述的情形時你總會得到 Accepted。如果你的程式被評為 **Accepted**，範例評分程式會分別輸出 T 次測試中呼叫 `activate_sonar_system()` 的次數。

如果你的程式被評為 **Wrong Answer**，範例評分程式輸出 Wrong Answer: MSG，其中 MSG 格式與意義如下：

- `too many queries!`：呼叫 `activate_sonar_system` 的次數超過 10 000 次。
- `invalid length`：傳入之聲納探測區間總長不滿足題目要求。
- `location(s) out of range`：傳入之快艇探測之 $p[i]$ 值超過允許範圍。
- `overlapping location`：傳入之快艇探測區間有所重疊。
- `incorrect number of clippers`：傳入之快艇位置數量不正確。
- `incorrect locations of debris`：回傳的金屬碎片位置不正確。
- `incorrect number of debris`：回傳的金屬碎片數量不正確。
- `corrupted stub`：可能改到 `stub` 了，請重新下載 `stub.cpp`。