Aufgabe Nette Linien

C header: `nice_lines_c.h`
C++ header: `nice_lines.h`

Roxette die Piratenprinzessin ist auf der geheimen Insel im Remeischen Archipelago angekommen. Laut Gerüchten ist dort ein berühmter Schatz, die *Goldenen netten Linien* vergraben.

Die geheime Insel ist ein Quadrat, 2×10^{12} mal 2×10^{12} lang und hoch. Jeder Punkt auf der Insel ist durch Kartesische Koordinaten beschrieben, wobei $(0, 0)$ die Mitte der Insel ist und die zwei Achsen parallel zu ihren Seiten verlaufen.

Es gibt N *Goldene nette Linien*, die auf der Insel vergraben sind. Für $0 \leq i < N$ bedeckt die i-te davon die Menge aller reellwertigen Punkte (x, y), beschrieben durch die lineare Gleichung $y = a_i x + b_i$.

Roxette kann ein spezielles Gerät verwenden, den *Lineometer*. Gegeben ein beliebiger Punkt p auf der Insel wird der *Lineometer* die Summe aller Distanzen[1] von Punkt p zu jeder der N *Goldenen netten Linien* bestimmen.

Leider erlaubt der *Lineometer* nur eine beschränkte Anzahl Verwendungen. Kannst du Roxette helfen, den Schatz mit einer zulässig kleinen Anzahl von *Lineometer*-Verwendungen zu finden?

Interaktionsprotokoll

Es gibt eine zu implementierende Funktion:

```
(C)   void solve(int subtask_id, int N);
(C++) void solve(int subtask_id, int N);
```

Diese Funktion wird **genau einmal** aufgerufen, am Anfang der Interaktion. N ist die Anzahl von *Goldenen netten Linien* die auf der Insel versteckt sind.

Diese Funktion kann eine andere Funktion aufrufen, aber **höchstens** $Q_{\mathbf{max}}$ Mal:

```
(C)   long double query(long double x, long double y);
(C++) long double query(long double x, long double y);
```

Diese Funktion muss mit Argumenten aufgerufen werden für die gilt: $-10^{12} \leq x, y \leq 10^{12}$.

Sie gibt das Resultat des *Lineometers* zurück, wenn dieses auf einen Punkt mit Kartesischen Koordinaten (x, y) angewendet wird – also die Summe der Distanzen von Punkt (x, y) zu jedem der N *Goldenen netten Linien*. Merke, dass die *Goldenen netten Linien* selbst nicht gegeben sind, da das Ziel ja gerade ist, diese zu finden.

Nachdem alle N *Goldenen netten Funktionen* gefunden sind, muss die folgende Funktion aufgerufen werden:

```
(C)   void the_lines_are(int* a, int* b);
(C++) void the_lines_are(std::vector<int> a, std::vector<int> b);
```

Wobei $a[i]$ und $b[i]$ die i-te *Goldene nette Linie* beschreiben müssen, für $0 \leq i < N$. Die Linien können in beliebiger Reihenfolge ausgegeben werden.

Limits

[1] Die Euklidsche Distanz zwischen einem Punkt und einer Linie ist die Länge des kürzesten Liniensegmentes welches sowohl die Linie wie auch den Punkt berührt.

- $1 \leq N \leq 100$
- $-10\ 000 \leq a_i, b_i \leq 10\ 000$
- Keine zwei Linien sind parallel.

Bewertung

Um die Punktzahl für einen Testfall zu berechnen, gehe wie folgt vor:

- Sei Q die Anzahl Aufrufe zur `query`-Funktion.
- Falls $Q > Q_{\max}$, oder falls die *Goldenen netten Linien* nicht korrekt ausgegeben wurden ist die Punktzahl für den Testfall 0.
- Falls $Q \leq Q_{\min}$, dann ist die Punktzahl für den Testfall 1.
- Andernfalls ist die Punktzahl für den Testfall $1 - 0.7 \cdot \frac{Q - Q_{\min}}{Q_{max} - Q_{\min}}$.

Um die Punktzahl für eine Teilaufgabe zu berechnen, nimm die kleinste Punktzahl die einem der Testfälle in dieser Teilaufgabe zugeordnet wurden und multipliziere sie mit der maximalen Punktzahl für diese Teilaufgabe.

Teilaufgabe 1 (11 Punkte)

- $N = 1$
- $Q_{\min} = 10\ 000$, $Q_{\max} = 10\ 000$

Teilaufgabe 2 (13 Punkte)

- $N = 2$
- $Q_{\min} = 10\ 000$, $Q_{\max} = 10\ 000$

Teilaufgabe 3 (7 Punkte)

- $N = 3$
- $Q_{\min} = 10\ 000$, $Q_{\max} = 10\ 000$

Teilaufgabe 4 (19 Punkte)

- $-500 \leq a_i, b_i \leq 500$
- $Q_{\min} = 402$, $Q_{\max} = 10\ 000$

Teilaufgabe 5 (23 Punkte)

- $N \leq 30$

- $Q_{\min} = 402$, $Q_{\max} = 10\ 000$

Teilaufgabe 6 (27 Punkte)

- $Q_{\min} = 402$, $Q_{\max} = 10\ 000$

Beispiel

Aufrufe des Kommittees	Aufrufe der Teilnehmenden

`solve(`
`/* subtask_id = */ 1,`
`/* N = */ 1)`
	`query(0, 0) gibt 0 zurück`
	`query(1, 1) gibt 0 zurück`
	`the_lines_are(`
	`/* a = */ {1},`
	`/* b = */ {0})`