เทศกาลตามหารัก (奉心祭)

By Leomotors

奉心祭 (Hoshinsai) มีต้นกำเนิดมาจากตำนานว่าด้วยเรื่องของชาย ผู้ยอมถวายดวงใจหัวใจ เพื่อพิสูจน์ความรัก

เทศกาลวัฒนธรรม (Bunkasai) ในปีนี้ที่จัดในโรงเรียนของคุณ ก็ได้มาในธีมของ Hoshinsai ซึ่งสัญลักษณ์เด่นก็คือหัวใจ มีคนจำนวนมากมายที่มีหวานใจอยู่แล้ว หรือบางคนก็กำลังใช้โอกาสนี้ในการ "*ตามหารัก*"

คุณก็เป็นคนหนึ่งที่ความปรารถนาที่จะ*ตามหารัก* ในงานเทศกาลวัฒนธรรมแห่งนี้ แต่ก็ดูท่าจะไปได้ไม่ค่อยสวยเท่าไหร่ จนได้พบกับเทพธิดาแห่งความรัก เทพธิดาองค์นี้เสนอที่จะช่วยคุณตามหา**รัก** แต่มีเงื่อนไขคือคุณต้องตามหาสิ่งที่คล้ายกันเป็นการตอบแทน นั่นคือ**ราก** (คล้ายกันตรงไหน?)

เทพธิดาองค์นั้นได้เล่าให้คุณฟังว่า ในโลกของคณิตศาสตร์ จะมีสิ่งที่เรียกว่าฟังก์ชันพหุนามอยู่ นั่นคือฟังก์ชันนั้นสามารถเขียนได้ในรูปของ

$f(x) = a_1x^n + a_2x^{n-1} + \cdots + a_nx + c$ เมื่อ $a_i \in \mathbb{R}$ และ $a_1 \neq 0$

และก็มีพหุนามบางตัวสามารถเขียนได้ในรูปของ

$f(x) = c \cdot (x - a_1)(x - a_2)\ldots(x - a_n)$

ซึ่งจะได้ว่าพหุนามนี้มีรากเป็น a_i โดยรากของพหุนามคือค่า x ที่ทำให้ $f(x) = 0$

งานของคุณ

คุณจะได้รับฟังก์ชัน ๆ หนึ่ง จากเทพธิดาแห่งความรัก ซึ่งมีลักษณะดังนี้

```
double f(double x)
```

โดยที่ฟังก์ชันดังกล่าว เป็นฟังก์ชันพหุนามที่มีดีกรี n และสามารถเขียนได้ในรูป

$f(x) = (x - a_1)(x - a_2)\dots(x - a_n)$

โดยงานของคุณคือการหาค่า x ใด ๆ ก็ได้ที่ทำให้ $f(x) = 0$ หรือก็คือหา x ที่ $x \in \{a_1, a_2, \dots, a_n\}$ โดยเรียกใช้ฟังก์ชัน f ให้เป็นจำนวนน้อยครั้งที่สุด

รายละเอียดการเขียนโปรแกรม

การเขียนโปรแกรมในข้อนี้จะเป็นรูปแบบ Library หรือก็คือรูปแบบเดียวกับการแข่งขัน IOI, สสวท และ TOI กำลังพิจารณาไปใช้ในปีหน้า ผู้เข้าแข่งขันจะเขียนเฉพาะฟังก์ชันเท่านั้น ไม่จำเป็นต้อง (และห้าม) เขียนส่วนที่เกี่ยวกับ I/O

ภายในไฟล์แนบจะมี `public.zip` ซึ่งจะประกอบไปด้วยเกรดเดอร์ตัวอย่างเพื่อทดลองบนเครื่องของตัวเอง ก่อนนำไปส่งในเกรดเดอร์จริง สำหรับคำสั่งที่ใช้ในการคอมไพล์จะถูกแจ้งใน CMS และคุณสามารถดูโค้ดตัวอย่างได้ในส่วนของตัวอย่าง

คุณจะต้องเขียนฟังก์ชันต่อไปนี้

```
int find_root(std::function<double(double)> f)
```

- ฟังก์ชันนี้จะถูกเรียก 100 ครั้ง (ยกเว้นปัญหาย่อยแรกจะถูกเรียก 5 ครั้ง) คุณจะต้องเขียนฟังก์ชันให้รองรับการเรียกใช้**มากกว่าหนึ่งครั้ง**
- จะต้องรีเทิร์นค่า x ใดก็ได้ที่ทำให้ $f(x) = 0$
- ฟังก์ชันนี้รับพารามิเตอร์หนึ่งตัวคือ f ซึ่งเป็นฟังก์ชัน คุณสามารถเรียกฟังก์ชันนี้โดยใช้การเรียก `f(x)` หรือจะส่งต่อไปในฟังก์ชันอื่นก็ได้ (ดูตัวอย่าง)
- คุณสามารถเรียกฟังก์ชัน f ได้ไม่เกิน $1\,000\,000$ ครั้งต่อการเรียก `find_root` หนึ่งครั้ง
- คุณสามารถเรียกฟังก์ชัน f ได้ด้วยจำนวนจริง (`double`) ในช่วง $[-10^9,\ 10^9]$

เงื่อนไข

- a_i (รากทุกตัว) อยู่ในช่วง $[10^{-9},\ 10^9]$ และเป็นจำนวนเต็ม
- ดีกรีของพหุนาม n จะมีค่าไม่เกิน 10 $(1 \le n \le 10)$
- เรา**ไม่สามารถรับประกัน**ได้ว่าค่าที่ได้จากการเรียก f จะมีความแม่นยำ (เนื่องจากเป็น `double`)
- ในเกรดเดอร์จริง ฟังก์ชัน f จะใช้เวลา $\mathcal{O}(n)$

ปัญหาย่อย

1. (1 คะแนน) พหุนามทุกตัวอยู่ในรูปของ $f(x) = (x - 69)^n$
2. (8 คะแนน) พหุนามทุกตัวเป็นสมการเส้นตรง
3. (11 คะแนน) พหุนามทุกตัวเป็นพาราโบลา
4. (22 คะแนน) พหุนามทุกตัวอยู่ในรูปของ $f(x) = (x - a)^n$
5. (13 คะแนน) รากทุกตัวของพหุนามทุกตัว อยู่ในช่วง $[-10,\ 10]$ และดีกรีไม่เกิน 4
6. (15 คะแนน) พหุนามมีดีกรีไม่เกิน 3
7. (19 คะแนน) พหุนามมีดีกรีไม่เกิน 6
8. (11 คะแนน) ไม่มีเงื่อนไขเพิ่มเติม

การให้คะแนน

ฟังก์ชัน f จะคอยเก็บข้อมูลว่าคุณเรียกใช้ฟังก์ชันนี้กี่ครั้ง เป้าหมายของคุณคือ พยายามเรียกใช้ให้น้อยครั้งที่สุด คะแนนของคุณจะขึ้นอยู่กับจำนวนครั้งที่คุณเรียกใช้ เทียบกับของกรรมการ

เมื่อ

- Q แทนจำนวนครั้งรวมที่คุณใช้จากทุกการเรียกฟังก์ชัน ทั้งนี้คุณจะต้อง**ตอบถูกทุกครั้ง** และหากมีครั้งไหนที่คุณเรียกมากกว่า $1\,000\,000$ ครั้ง คะแนนที่คุณจะได้ในชุดทดสอบนี้คือ 0
- Q' แทนจำนวนครั้งรวมที่กรรมการใช้ ซึ่งจะถูกแจ้งทั้งในไฟล์แนบ `judge_query.zip` รวมถึงข้อความผลลัพธ์การตรวจ

คะแนนของคุณที่จะได้ในปัญหาย่อยนั้น ๆ คือ

เงื่อนไข	อัตราส่วนของคะแนนที่ได้ต่อคะแนนเต็มของปัญหาย่อยนั้น ๆ
$Q \le \lceil 1.03 * Q' \rceil$	1
$\lceil 1.03 * Q' \rceil < Q$	$\dfrac{2}{1 + \frac{Q}{Q'}}$

ตัวอย่าง

สมมติว่า $f(x) = x^2 - 3x + 2$ และมีการเรียก

ตัวอย่างการเขียนโค้ด

```
#include "find_root.h"
#include <functional>

int find_root(std::function<double(double)> f) {
    int a = f(-1);
    std::cout << a << "\n"; //prints 6
    int b = f(0);
    std::cout  << b << "\n"; // prints 2
    int c = f(1);
    std::cout << c << "\n"; // prints 0
    return 1; // Correct Answer
}
```

หมายเหตุ สำหรับใครที่สงสัยว่าทำไมถึงให้ include `find_root.h` นั่นเพราะตอนแรกโจทย์ชื่อนี้ แล้วแอดมินขี้เกียจเปลี่ยน

จะมีการเรียก

```
find_root(f)
```

ภายในฟังก์ชันมีการเรียกใช้ `f(-1)` ซึ่งคืนค่า 6 จากนั้นจึงเรียกใช้ `f(0)` ซึ่งคืนค่า 2 และต่อมาเรียก `f(1)` ซึ่งคืนค่า 0 ทำให้เราทราบว่า 1 เป็นหนึ่งในรากของ f หรือ $f(1) = 0$ ดังนั้นจึงรีเทิร์น 1 ซึ่งเป็นคำตอบที่ถูกต้อง อีกหนึ่งคำตอบที่สามารถตอบได้เช่นกันคือ 2 เนื่องจาก $f(2) = 0$ เหมือนกัน

หมายเหตุ ในการตรวจจริง **ห้าม**โค้ดของผู้เข้าแข่งขัน ใช้คำสั่งเกี่ยวกับ IO (เช่น `printf`, `scanf`) เด็ดขาด ข้อห้ามนี้รวมถึงข้ออื่นที่เป็น Library ด้วย

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างจะอ่านข้อมูลนำเข้าดังนี้

- บรรทัดที่ 1: T แสดงจำนวนชุดทดสอบ
- บรรทัดที่ $1+i$ ถึง $1+T$: $n\ \ a_1\ \ a_2\ \ \ldots\ a_n$

เกรดเดอร์ตัวอย่างจะพิมพ์ $T+1$ บรรทัด โดยบรรทัดที่ i จนถึง T จะแสดงจำนวนของ Q ที่คุณใช้ในชุดทดสอบที่ i และบรรทัดที่ $T+1$ จะพิมพ์ผลรวมของ Q ซึ่งจะเป็นตัวที่ถูกใช้ในการคิดคะแนน

ขีดจำกัด

- Time limit: 3 seconds
- Memory limit: 1024 MB