Mastermind

0.1 second (รันฟังก์ชันหลายรอบ), 512 MB, interactive

เกม mastermind เป็นเกมที่เล่นสองคน โดยผู้เล่นคนหนึ่งจะเป็นคนตั้งโจทย์ ส่วนอีกคนเป็นผู้ทาย ในการเล่นจะมีการเลือกพารามิเตอร์สามค่าคือ N จำนวนหลัก K จำนวนตัวเลขที่ใช้ได้ในแต่ละหลัก (ไม่เกิน 9) และ U ที่บอกว่าตัวเลขในโจทย์มีการซ้ำกันได้หรือไม่ (เป็น 1 หรือ 0) พารามิเตอร์นี้จะทราบทั้งผู้ตั้งโจทย์และผู้ทาย

ก่อนเริ่มเกม ผู้ตั้งโจทย์จะเลือกเลขเฉลย N หลัก ที่แต่ละหลักมีค่าตั้งแต่ 1 – K โดยถ้า U = 0 อาจจะมีเลขซ้ำได้ ถ้า U = 1 จะไม่มีเลขซ้ำ เรียกตัวเลขนี้เป็น S และจะเก็บเป็นความลับ

เมื่อเริ่มเกม ในแต่ละรอบ ผู้ทายจะส่งเลข N หลัก ที่แต่ละหลักมีค่าตั้งแต่ 1 – K โดยอาจจะมีเลขซ้ำได้ ให้กับผู้ตั้งโจทย์ เรียกตัวเลขนี้เป็น g ผู้ตั้งโจทย์จะให้คะแนนเป็นจำนวนเต็มสองตัวคือ

1. จำนวนหลักที่ตรง (num_matched) คือจำนวนหลักใน g ที่มีค่าตรงกับหลักเดียวกันใน S
2. จำนวนหลักที่ไม่ตรง (num_misplaced) จำนวนตัวเลขที่เหลือที่ตรงกัน (นับซ้ำด้วย) เมื่อหักหลักต่าง ๆ ที่ตรงกันในข้อ 1 ออกแล้ว

พิจารณาตัวอย่างการให้คะแนนดังนี้ (ในตัวอย่าง N = 4, K = 4)

เฉลย S	เลขที่ทาย g	คะแนน	คำอธิบายคะแนนตัวที่สอง
1234	1324	2 2	{2,3} & {2,3}
1234	1321	1 2	{2,3,4} & {3,2,1}
1233	1321	1 2	{2,3,3} & {3,2,1}
1233	1333	2 1	{2,3} & {3,3}
1233	3333	1 1	{1,2,3} & {3,3,3}
1233	3322	0 3	{1,2,3,3} & {3,3,2,2}

คุณจะต้องเขียนฟังก์ชัน find_answer(int N, int K, int U) ที่เล่นเกมนี้ โดยในฟังก์ชันดังกล่าว คุณสามารถเรียกใช้ฟังก์ชัน guess เพื่อทายได้ คุณต้องพยายามทายโดยใช้จำนวนครั้งที่น้อยที่สุด ฟังก์ชัน guess มีรูปแบบดังนี้

guess(string g, int& num_matched, int& num_misplaced)

คุณจะต้องส่งสตริง g และจะได้คำตอบคืนมาในตัวแปร num_matched และ num_misplaced เมื่อคุณทายได้ถูกหมด (เมื่อ num_matched = N) ให้ฟังก์ชัน find_answer จบการทำงานด้วยการ return (เกรดเดอร์จะไม่จบการทำงานให้เอง) ถ้าคุณยังถามต่อคุณจะผิดทันที

ฟังก์ชัน find_answer จะถูกเรียกใช้หลายรอบระหว่างการตรวจ รับประกันว่าจะไม่เรียกเกิน 1000 ครั้ง เวลาการทำงานรวมของโปรแกรมของคุณทั้งหมดและโปรแกรมเกรดเดอร์จะต้องไม่เกิน time limit ของข้อนี้ เวลาที่เกรดเดอร์ใช้นั้นน้อยมากจนคุณสามารถลืมมันไปได้เลย

ตัวอย่างการทำงาน

ด้านล่างเป็นตัวอย่างการเล่นที่ N = 2, K = 3, U = 1 สมมติว่า S="13"

ฟังก์ชันของคุณจะถูกเรียกด้วย

find_answer(2, 3, 1)

ด้านล่างเป็นตัวอย่างการเรียกใช้ guess

คำสั่งที่เรียก	num_matched	num_misplaced
guess("12",num_matched,num_misplaced)	1	0
guess("32",num_matched,num_misplaced)	0	1
guess("13",num_matched,num_misplaced)	2	0

เมื่อคุณทายได้ถูกหมด (เมื่อ num_matched = N) ให้ฟังก์ชันจบการทำงาน (เกรดเดอร์จะไม่จบการทำงานให้เอง)

การทดลองเขียนและเรียกทำงาน

ในการทดลองเรียกทำงาน ให้คุณเขียนฟังก์ชัน find_answer ในไฟล์ mastermind-play.cpp และคอมไพล์ร่วมกับ mastermind-lib.cpp

ข้อมูลนำเข้าสำหรับการทดสอบอยู่ในรูปแบบ

N K U

S

การให้คะแนน ปัญหานี้จะแบ่งเป็น subtask หลายอัน แต่ละ subtask จะมีช่วงของค่า N K และ U อยู่ แต่ละ subtask อาจจะมีหลายเทสรัน แต่ละเทสรันจะระบุค่า N K และ U ในช่วงของ subtask แต่ละเทสรัน โปรแกรมของคุณจะถูกทดสอบกับเลขเฉลยทุกรูปแบบ คุณจะได้คะแนนเต็มสำหรับเทสรันนั้น ถ้าจำนวนครั้งที่แย่ที่สุดที่คุณใช้ในการถาม ไม่เกินค่าที่น้อยที่สุดที่สามารถทำได้ (รับประกันว่าทำได้) คุณจะได้ 10% ถ้าโปรแกรมคุณทำงานทันและตอบถูกต้องแต่มีการถามที่มากเกินไป คุณจะเห็นผลลัพธ์ของการตรวจทันที คะแนนของ subtask จะเป็นคะแนนของเทสรันใน subtask นั้นที่ต่ำที่สุด คะแนนของแต่ละ subtask ของคุณจะเป็นคะแนนสูงสุดในแต่ละ submission

ด้านล่างเป็นตารางสรุปค่า N/K ที่มีการทดสอบ

N\K	2	3	4	5	6	7	8	9
1	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x
3	x	x	x	x	x			
4	x	x	x					
5	x	x						
6	x							
7	x							

(ขอบเขตของแต่ละ subtask จะอยู่หน้าถัดไป)

Subtask	ขอบเขต
1	N=1 (ทุก K, U=0)
2	N=2 (ทุก K, ทุก U)
3	N=3 (ทุก K, ทุก U)
4	N=4,5,6,7 (ทุก K, ทุก U)
5	K=2 (ทุก N ทุก U)
6	K=3 (ทุก N ทุก U)
7	K=4 (ทุก N, U = 0)
8	K=4 (ทุก N, U = 1)
9	K=5,6 (ทุก N, U = 0)
10	K=5,6 (ทุก N, U = 1)
11	K=7,8,9 (ทุก N, U = 0)
12	K=7,8,9 (ทุก N, U = 1)