บีเวอร์ (beaver)

คอมพิวเตอร์บีเวอร์ประมวลผลข้อมูลที่อยู่บนอาร์เรย์ของบิตแบบ 1 มิติที่ (ในทางทฤษฎี) มีความยาวไม่จำกัด และเป็นอาร์เรย์แบบสองทาง นั่นคือในการอ้างค่าในอาร์เรย์สามารถใช้ดัชนีที่เป็นจำนวนเต็มใด ๆ ก็ได้ (นั่นคือมีดัชนีที่เป็นค่าลบได้) เมื่อเริ่มต้นทุกช่องในอาร์เรย์จะมีค่าเป็น `0`

ในขณะใด ๆ คอมพิวเตอร์จะเก็บดัชนี (หรือมองว่าเป็น data pointer ก็ได้) ไปยังช่องช่องหนึ่งในอาร์เรย์ เมื่อเริ่มต้นค่าดัชนี (data pointer) จะเป็น 0 เพื่อให้เห็นภาพ เราอาจจะพิจารณาว่าตัวบีเวอร์นั้นอยู่บนอาร์เรย์ที่ช่องที่ data pointer ชี้อยู่ก็ได้

โปรแกรมภาษาบีเวอร์คือสตริงที่ประกอบไปด้วยอักขระ `<`, `>`, `(`, `)`, และ `*` และมีการทำงานบนคอมพิวเตอร์บีเวอร์ไปตามลำดับดังนี้ ถ้าคำสั่งที่ทำงานคือ:

คำสั่ง	การทำงาน
คำสั่ง `<`	จะลด data pointer ลงหนึ่ง (บีเวอร์เดินไปทางซ้าย)
คำสั่ง `>`	จะเพิ่ม data pointer ขึ้นหนึ่ง (บีเวอร์เดินไปทางขวา)
คำสั่ง `*`	จะสลับบิตในอาร์เรย์ช่องที่ data pointer ชี้อยู่ (บีเวอร์สลับบิต)

คำสั่ง `(` และ `)` จะเกี่ยวข้องกับการควบคุมลำดับการทำงานในโปรแกรม ในการกระโดดไปมาในโปรแกรมจะใช้ตำแหน่งของวงเล็บเปิด `(` และปิด `)` ที่จับคู่กันเป็นหลัก

คำสั่ง	การทำงาน
คำสั่ง `(`	จะพิจารณาค่าในอาร์เรย์ช่องที่ data pointer ชี้อยู่ ถ้าค่านั้นเป็น `0` จะกระโดดไปทำงานที่คำสั่งถัดไปจากคำสั่งวงเล็บปิด `)` ที่จับคู่กับวงเล็บนี้ ถ้าค่าเป็น `1` จะทำงานตามคำสั่งถัดไปตามปกติ
คำสั่ง `)`	จะพิจารณาค่าในอาร์เรย์ช่องที่ data pointer ชี้อยู่ ถ้าค่านั้นเป็น `1` จะกระโดดไปทำงานที่คำสั่งถัดไปจากคำสั่งวงเล็บปิด `(` ที่จับคู่กับวงเล็บนี้ ถ้าค่าเป็น `0` จะทำงานตามคำสั่งถัดไปตามปกติ

ในโปรแกรมสามารถมีคำสั่ง `(` และ `)` ได้หลายคำสั่ง และสามารถมีคำสั่งเหล่านี้ซ้อนกันได้ ในการตีความจะใช้การจับคู่กันของวงเล็บตามมาตรฐานคณิตศาสตร์ ในกรณีที่วงเล็บไม่มีการจับคู่ที่ถูกต้อง ถ้าไม่มีการกระโดดบีเวอร์ก็จะทำงานต่อไปตามปกติ ถ้ามีการกระโดดแต่วงเล็บที่จับคู่กันไม่มี จะมีการฟ้องความผิดพลาดออกมา

คอมพิวเตอร์บีเวอร์จะทำงานตามคำสั่งไปเรื่อย ๆ จนหมดทุกคำสั่งในโปรแกรม

พิจารณาตัวอย่างการทำงานของโปรแกรมบีเวอร์ง่าย ๆ ดังนี้

- `<<*>>>*` ในโปรแกรมนี้บีเวอร์จะเดินไปทางซ้ายสองก้าว เปลี่ยนบิตจาก `0` เป็น เป็น `1` จากนั้นเดินขวาสามก้าว และเปลี่ยนบิตจาก `0` เป็น `1` ผลลัพธ์บนอาร์เรย์จะเป็นดังนี้ `...10`**`0`**`1...` โดยช่องที่ดัชนี `0` คือช่องที่เป็นตัวเข้ม ช่องที่บีเวอร์ไม่ได้แตะแสดงเป็นจุด
- `*()` จะเป็นโปรแกรมที่ทำงานไม่สิ้นสุด เพราะว่าบีเวอร์เปลี่ยนอาร์เรย์ช่องปัจจุบันให้เป็น `1` จากนั้นก็จะวนลูปไปมาไม่รู้จบ แต่บีเวอร์จะอยู่ที่ช่องดัชนี `0` ตลอด
- `*(*)` โปรแกรมนี้ทำงานได้สี่ขั้นตอนแล้วจบการทำงาน เพราะว่าบิตในอาร์เรย์ช่องที่ `0` ถูกเปลี่ยนเป็น `0` อีกครั้งหลังคำสั่ง `*` ที่สอง ทำให้บีเวอร์ไม่กระโดดกลับไปที่ท้ายคำสั่ง `(`
- `*(>*)` โปรแกรมนี้ก็ทำงานไม่สิ้นสุดเช่นเดียวกัน แต่บีเวอร์จะเดินไปทางขวาเรื่อย ๆ ด้วย
- `*((*)*)` โปรแกรมนี้ก็ทำงานไม่สิ้นสุด บีเวอร์จะทำงานเข้าไปถึงคำสั่ง `*` ที่สอง จากนั้นจะเปลี่ยนบิตในอาร์เรย์ช่องที่ `0` กลับไป `0` ทำให้คำสั่งวงเล็บปิดแรกไม่มีการกระโดด แต่คำสั่ง `*` ที่สาม สลับบิตกลับเป็น `1` อีกครั้ง ทำให้คำสั่งวงเล็บปิดที่สองบังคับให้กลับไปทำงานต่อจากคำสั่งวงเล็บเปิดอันแรก

การเขียนโปรแกรมให้ทำงานจบแบบรวดเร็วนั้นง่ายดาย การเขียนโปรแกรมให้ทำงานไม่สิ้นสุดนั้นก็ดูจะไม่ยากขึ้นสักเท่าใด แต่การเขียนโปรแกรมให้ทำงานสิ้นสุดด้วย แต่ทำงานเป็นระยะเวลายาวนานนั้นน่าจะเป็นสิ่งที่ท้าทาย และนั่นคืองานของคุณในข้อนี้

งานของคุณ คือการเขียนโปรแกรมภาษาบีเวอร์ที่ทำให้บีเวอร์เดินทางไป *สัมผัสกับจำนวนช่องต่าง ๆ ให้ได้จำนวนช่องมากที่สุดเท่าที่เป็นไปได้* ดูเพิ่มเติมในส่วน **การให้คะแนน**

หมายเหตุ เนื่องจากเราต้องตรวจโปรแกรมบีเวอร์ของคุณบนคอมพิวเตอร์จริง ๆ ทำให้เราไม่สามารถปล่อยโปรแกรมของคุณทำงานไปได้เรื่อย ๆ และเราไม่สามารถ**ตัดสิน**ได้ด้วยว่าโปรแกรมของคุณจะทำงานไม่สิ้นสุดหรือไม่ ดังนั้น ข้อนี้จึงมีข้อจำกัดว่าเราจะจำลองการทำงานของโปรแกรมบีเวอร์ไปไม่เกิน $2\,000\,000\,000$ ขั้นตอนเท่านั้น ถ้าเกินค่านี้เราจะถือว่าโปรแกรมบีเวอร์ของคุณทำงานไม่รู้จบ

การให้คะแนน

หาก S แทนจำนวนช่องที่คุณไปถึง คุณจะได้รับคะแนนตามตารางดังต่อไปนี้

เงื่อนไข	คะแนน
$1 \le S \le 50$	5
$51 \le S \le 23\,100$	$6.9 \log_2 (S - 49)$
$23\,101 \le S$	100

รายละเอียดการเขียนโปรแกรม

สำหรับข้อนี้ จะใช้ระบบคล้าย output-only กล่าวคือ คุณจะต้องส่งไฟล์ plain text `output_01.txt` มายังระบบตัวตรวจ เงื่อนไขสำคัญคือไฟล์ดังกล่าวจะต้องมีข้อมูล **ไม่เกิน 49 ตัวอักษร** ประกอบด้วยข้อมูลบรรทัดเดียวที่ระบุโปรแกรมภาษาบีเวอร์ที่ถูกต้อง

ตัวตรวจอัตโนมัติอาจรับข้อมูลมากกว่า 49 ตัวอักษรสำหรับอักขระประเภท whitespace อย่างไรก็ตาม ตัวตรวจอัตโนมัติจะทำการละทิ้งอักขระเหล่านี้และพิจารณาเพียงโปรแกรมภาษาบีเวอร์เท่านั้น

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่าง (`autograder.cpp`) จะอ่านข้อมูลนำเข้าบรรทัดเดียว ระบุโปรแกรมภาษาบีเวอร์ และจะส่งออกข้อมูลส่งออก 3 บรรทัด ดังนี้

- บรรทัดแรก `Number of reached cells:` S แทนจำนวนช่องที่โปรแกรมภาษาบีเวอร์ได้ทำการเดินไปถึง
- บรรทัดที่สอง `Instruction count:` แทนจำนวนคำสั่งที่ใช้ หากเกิน $2\,000\,000\,000$ จะมีการแจ้งเตือนและยุติการทำงาน
- บรรทัดที่สาม `Score:` แทนคะแนนที่คาดว่าจะได้รับหากทำการส่งโปรแกรมเข้ามายังระบบตรวจ