Rescue (ช่วยคนติดถ้ำ)

Time limit: 1 sec *memory limit: 512mb*

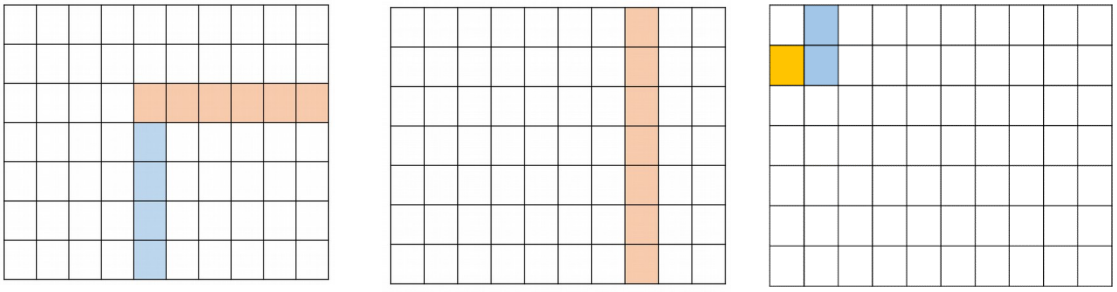
ติดถ้ำ!!! นายอีธาน ประสบภัยพิบัติติดถ้ำอยู่ในห้องโถงใหญ่ ห้องโถงนี้เป็นพื้นที่สี่เหลี่ยมขนาดกว้าง C หน่วยและยาว R หน่วย โดยที่ 1 <= C,R <= 1,000 ตำแหน่งต่าง ๆ ในห้องนี้สามารถระบุด้วยพิกัด (r, c) โดยที่ 1 <= r <= R และ 1 <= c <= C การที่จะช่วยเหลืออีธานออกมาให้ได้นั้นเราจะต้องระบุตำแหน่งที่แน่นอนของอีธานให้ได้เสียก่อน

เรามีหุ่นยนต์จิ๋วแบบใช้แล้วทิ้งอยู่หลายตัว เราสามารถทิ้งหุ่นยนต์ดังกล่าวลงไปที่พิกัดใด ๆ ก็ได้ เมื่อทิ้งหุ่นยนต์ดังกล่าวแล้ว หุ่นยนต์จะวิ่งไปหาอีธานด้วยเส้นทางที่สั้นที่สุด โดยหุ่นยนต์สามารถเดินทางจากช่อง (r, c) ใด ๆ ไปยังช่องที่มีด้านติดกับช่อง (r, c) ได้เท่านั้น (กล่าวคือ สามารถเดินได้ในทิศ บน ล่าง ซ้าย ขวา เท่านั้น) โดยหุ่นยนต์ไม่สามารถเดินทางทะลุกำแพงใด ๆ ได้ เมื่อหุ่นยนต์วิ่งไปจนถึงพิกัดที่อีธานอยู่แล้ว หุ่นยนต์จะส่งสัญญาณกลับมาหาเรา อย่างไรก็ตาม เนื่องจากการติดต่อสื่อสารเป็นไปได้อย่างยากลำบาก หุ่นยนต์จึงส่งข้อมูลกลับมาเพียงว่า หุ่นยนต์ต้องเดินทางไปเป็นระยะทางกี่ช่องถึงจะถึงอีธาน เมื่อส่งข้อมูลแล้ว หุ่นยนต์จะหยุดทำงาน

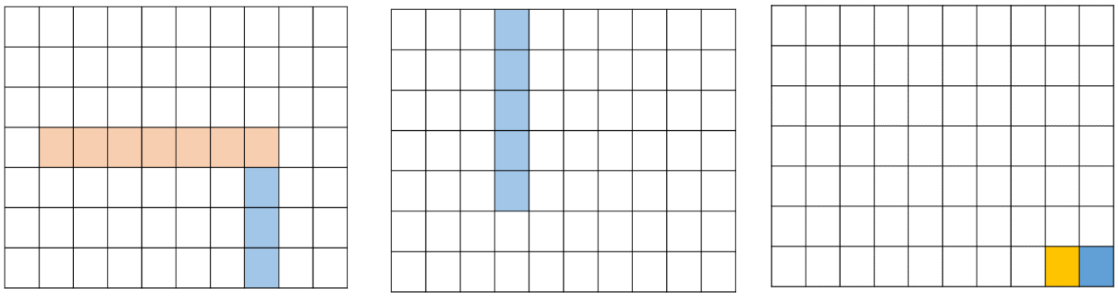
ห้องโถงใหญ่นี้อาจจะมีกำแพงอยู่เป็นจำนวน 0, 1 หรือ 2 กำแพง กำแพงแต่ละอันจะเรียงตัวเป็นเส้นตรงตามแนวกว้างหรือแนวยาวเสมอ ถ้าหากมีกำแพงเหล่านี้จะแบ่งห้องโถงของเราออกเป็นสองห้องย่อยเสมอ กล่าวคือ

- ในกรณีที่มี 1 กำแพง กำแพงจะมีปลายกำแพงชนกับขอบของห้องโถงเสมอ
- ในกรณีที่มี 2 กำแพง กำแพงแต่ละอันจะมีปลายด้านหนึ่งชนกับขอบของห้องโถง และปลายอีกด้านหนึ่งสัมผัสกันเอง และกำแพง 1 กำแพงจะอยู่ในแนวกว้าง และอีกกำแพงอยู่ในแนวยาว
- ในกรณีที่มีกำแพง รับประกันว่า กำแพงจะกั้นให้มีห้องที่มีพื้นที่อย่างน้อย 1 ช่องอยู่เสมอ และอีธานจะไม่อยู่ในช่องที่เป็นเนื้อกำแพงแน่นอน

ตัวอย่างด้านล่างนี้แสดงถึงรูปแบบกำแพงที่เป็นไปได้ ของห้องโถงที่กว้าง 10 หน่วย ยาว 7 หน่วย



ตัวอย่างด้านล่างนี้ แสดงถึง**<u>กำแพงที่เป็นไปไม่ได้</u>**



เราต้องการหาตำแหน่งของอีธาน โดยใช้จำนวนหุ่นยนต์ให้น้อยที่สุด

โจทย์ข้อนี้เป็นแบบ ถาม-ตอบ คุณสามารถเรียกฟังก์ชันต่อไปนี้ในการทำงานได้

- void get_size(int &R, int &C) เป็นฟังก์ชันเพื่อถามหาขนาดความกว้างและความยาวของห้องโถงใหญ่ โดยเราจะต้องเรียกฟังก์ชันนี้ก่อนการเรียกใช้ฟังก์ชันอื่น ๆ รับประกันว่า 0 < R,C <= 1,000

- int drop_robot(int r, int c) จะเป็นการทิ้งหุ่นยนต์ลงไปที่พิกัด (r, c) ฟังก์ชันนี้จะคืนค่าระยะทางที่หุ่นยนต์ต้องเดินทางไปหาอีธาน ถ้าหากหุ่นยนต์ไม่สามารถเดินไปได้ หรือหุ่นยนต์ถูกทิ้งไปยังตำแหน่งที่เป็นกำแพง ฟังก์ชันนี้จะคืนค่า -1 การเรียกฟังก์ชันนี้จะต้องตรงตามเงื่อนไข 1 <= r <= R และ 1 <= c <= C ไม่เช่นนั้นจะถือว่าโปรแกรมทำงานผิดพลาด
- void answer(int r, int c) ให้เรียกฟังก์ชันนี้เพื่อระบุว่าอีธานอยู่ ณ ตำแหน่ง (r, c) เมื่อเรียกฟังก์ชันนี้แล้ว โปรแกรมจะหยุดทำงาน

โปรแกรมของคุณจะต้องติดต่อกับ library โดยให้ #include "rescues.h" ที่ต้นโปรแกรมและในตอนคอมไพล์ให้นำ rescuelib.cpp ไปคอมไฟล์ด้วย ห้ามโปรแกรมทำการอ่านเขียนข้อมูลเอง

Subtask

ในแต่ละข้อมูลทดสอบนั้นจะได้คะแนนก็ต่อเมื่อเรียก answer เป็นจำนวน 1 ครั้งพอดีด้วยคำตอบที่ถูกต้อง และ จำนวนครั้งที่เรียก drop_robot จะต้องไม่เกินเงื่อนไขที่กำหนด

สำหรับปัญหาย่อยที่ 2 – 6 นั้นจะมีข้อมูลทดสอบอื่นที่ไม่ได้เปิดเผยระหว่างทำการสอบมาคิดคะแนนด้วย โดยมีอัตราส่วนของข้อมูลที่ไม่ได้เปิดเผยไม่เกิน 50% ของแต่ละข้อมูลชุดทดสอบ

- ปัญหาย่อยที่ 1 (10%): ไม่มีกำแพงใด ๆ ห้ามเรียก drop_robot เกิน 1,000,000 ครั้ง
- ปัญหาย่อยที่ 2 (10%): ไม่มีกำแพงใด ๆ ห้ามเรียก drop_robot เกิน 100 ครั้ง
- ปัญหาย่อยที่ 3 (10%): ไม่มีกำแพงใด ๆ ห้ามเรียก drop_robot เกิน 4 ครั้ง
- ปัญหาย่อยที่ 4 (20%): มีกำแพงจำนวน 1 กำแพง ห้ามเรียก drop_robot เกิน 60 ครั้ง
- ปัญหาย่อยที่ 5 (10%): มีกำแพงจำนวน 1 กำแพง ห้ามเรียก drop_robot เกิน 4 ครั้ง
- ปัญหาย่อยที่ 6 (40%): มีกำแพงจำนวน 2 กำแพง ห้ามเรียก drop_robot เกิน 30 ครั้ง

Example Library

ไลบรารี่ตัวอย่างจะอ่านข้อมูลนำเข้าจาก Keyboard ตามรูปแบบต่อไปนี้

- บรรทัดแรกประกอบด้วยจำนวนเต็มสองตัวระบุความกว้างและความยาวของห้อง
- บรรทัดที่ 2 ประกอบด้วยจำนวนเต็มสองตัวระบุตำแหน่งของอีธาน
- บรรทัดที่ 3 ประกอบด้วยจำนวนเต็ม M ซึ่งระบุจำนวนกำแพง
- หลังจากนั้นอีก M บรรทัดเป็นข้อมูลกำแพง แต่ละบรรทัดประกอบด้วยจำนวนเต็ม 4 ตัวคือ r1 c1 r2 c2 ซึ่งระบุว่ามีกำแพงเป็นแนวเส้นตรงเชื่อมพิกัด (r1, c1) กับ (r2, c2) โดยที่ r1 == r2 หรือไม่ก็ c1 == c2

ไลบรารี่ในเกรดเดอร์จะทำงานแตกต่างจากไลบรารีตัวอย่าง แต่จะมีฟังก์ชันให้เรียกใช้งานได้เหมือนกัน