International Olympiad in Informatics 2012
23-30 September 2012
Sirmione - Montichiari, Italy
Competition tasks, day 2: Leonardo's art and science

supper
ไทย — 1.2

อาหารค่ำมื้อสุดท้าย

ในช่วงที่ลีโอนาร์โดกำลังขะมักเขม้นเขียนภาพอาหารค่ำมื้อสุดท้าย (The Last Supper) ซึ่งเป็นจิตรกรรมฝาผนังที่มีชื่อเสียงมากที่สุดของเขา งานประจำในทุก ๆ วัน คือ เขาต้องตัดสินใจว่าจะใช้สีฝุ่นสีใดสำหรับการทำงานในวันนั้น เขาต้องการสีฝุ่นจำนวนหลายสีแต่บนนั่งร้านของเขาเก็บสีได้จำนวนจำกัด ผู้ช่วยของเขาจึงต้องทำหน้าที่ปีนนั่งร้านเพื่อนำสีไปส่งให้ลีโอนาร์โด แล้วเอาสีลงมาเก็บบนชั้นเก็บสีซึ่งวางอยู่บนพื้น

ในโจทย์ข้อนี้ คุณจะต้องเขียนโปรแกรมสองโปรแกรมแยกกันเพื่อช่วยเหลือผู้ช่วย โปรแกรมแรกจะรับคำสั่งของลีโอนาร์โด (ลำดับของสีที่ลีโอนาร์โดจะต้องการใช้ระหว่างวัน) และสร้างสตริงของบิตที่ สั้น ซึ่งเรียกว่า คำแนะนำ ในขณะที่ประมวลผลคำร้องขอของลีโอนาร์โดระหว่างวัน ผู้ช่วยจะไม่ทราบถึงคำร้องขอในอนาคตของลีโอนาร์โด แต่จะอ่านเฉพาะคำแนะนำที่สร้างขึ้นจากโปรแกรมแรกของคุณ โปรแกรมที่สองจะรับคำแนะนำ รับและประมวลผลคำร้องขอของลีโอนาร์โดในแบบออนไลน์ (หนึ่งการร้องขอต่อหนึ่งครั้ง) โปรแกรมนี้จะต้องสามารถเข้าใจความหมายของคำแนะนำและใช้มันเพื่อสร้างตัวเลือกที่เหมาะสุด ตามคำอธิบายด้านล่าง

การย้ายสีระหว่างชั้นเก็บสีและนั่งร้าน

เราจะลองพิจารณาตัวอย่างอย่างง่าย โดยสมมติว่า มีสีจำนวน N สี ซึ่งกำกับด้วยหมายเลขตั้งแต่ 0 ถึง $N - 1$ และในแต่ละวัน ลีโอนาร์โด จะขอสีใหม่จากผู้ช่วย จำนวน N ครั้งพอดี กำหนดให้ C เป็นลำดับของการขอสีจำนวน N ครั้งจากลีโอนาร์โด ดังนั้นเราอาจคิดให้ C เป็นลำดับของตัวเลข N ตัว ซึ่งมีค่าตั้งแต่ 0 ถึง $N - 1$ (รวมหัวท้าย) ขอให้สังเกตว่า ในลำดับ C นั้น อาจไม่มีสีบางสีอยู่เลยก็เป็นได้ และอาจมีสีบางสีปรากฏอยู่มากกว่าหนึ่งครั้งก็ได้

นั่งร้านจะเต็มไปด้วยสีจำนวน K สีจาก N สีเสมอ โดยมีเงื่อนไขว่า $K < N$ และในตอนเริ่มต้นนั่งร้านจะมีสีหมายเลข 0 ถึง $K - 1$ (รวมหัวท้าย) อยู่แล้ว

ผู้ช่วยของเขาจัดการการร้องขอสีของลีโอนาร์โดครั้งละหนึ่งสี เมื่อใดก็ตามที่สีที่ได้รับการร้องขอ อยู่บนนั่งร้านแล้ว ผู้ช่วยจะสามารถพักผ่อนได้ ไม่เช่นนั้นแล้ว เขาจะต้องไปหยิบสีที่ลีโอนาร์โดขอจากชั้นเก็บสี และนำไปไว้บนนั่งร้าน แต่นั่งร้านนั้นเต็มอยู่ตลอดเวลาทำให้ไม่มีที่สำหรับสีใหม่ ดังนั้นผู้ช่วยต้องเลือกหยิบสีใดสีหนึ่งออกจากนั่งร้าน และเอากลับไปเก็บยังชั้นเก็บสี

กลยุทธเหมาะสุดของลีโอนาร์โด

ผู้ช่วยของลีโอนาร์โดต้องการพักให้มากครั้งที่สุดเท่าที่จะทำได้ จำนวนการร้องขอสีที่ทำให้เขาพักได้จะขึ้นกับการเลือกของเขาระหว่างการทำงาน หรือกล่าวให้ชัดเจนขึ้นว่า ในแต่ละครั้งที่ผู้ช่วยต้องย้ายสีออกจากนั่งร้าน การเลือกที่แตกต่างกันอาจจะทำให้ผลลัพธ์ในอนาคตต่างกัน ลีโอนาร์โดอธิบายวิธีที่จะทำให้ผู้ช่วยของเขาได้พักมากที่สุดเมื่อรู้ลำดับ C ว่า การเลือกสีออกจากนั่งร้านที่ดีที่สุดนั้นสามารถคำนวณได้จากสีที่อยู่บนนั่งร้าน และการร้องขอสีที่เหลือใน C โดยสีที่จะถูกเลือกออกจากนั่งร้านนั้นควรจะต้องเป็นไปตามกฎดังต่อไปนี้

- ถ้ามีสีที่อยู่บนนั่งร้านที่ไม่ต้องการใช้อีกแล้วในอนาคต ผู้ช่วยควรนำสีนี้ออกไปจากนั่งร้าน
- ไม่เช่นนั้น สีที่ถูกหยิบออกจากนั่งร้านควรเป็นสีที่ จะถูกเรียกใช้ครั้งต่อไปช้าสุดในอนาคต (นั่นคือ สำหรับแต่ละสีที่อยู่บนนั่งร้าน เราหาการปรากฏขึ้นในอนาคตเป็นครั้งแรกของสีเหล่านั้น สีที่ถูกนำกลับไปเก็บที่ชั้นเก็บสีจะเป็นสีที่เป็นที่ต้องการท้ายสุด)

มันสามารถพิสูจน์ได้ว่า หากเราใช้กลยุทธเหมาะสุดของลีโอนาร์โดแล้ว ผู้ช่วยจะได้พักเป็นจำนวนครั้งที่มากที่สุดเท่าที่จะเป็นไปได้

ตัวอย่างที่ 1

กำหนดให้ N = 4 ดังนั้นเราจะมีสี 4 สี (กำกับด้วยตัวเลขตั้งแต่ 0 ถึง 3) และมีการร้องขอ 4 ครั้ง ให้ลำดับของการร้องขอ C = (2, 0, 3, 0) และสมมติว่า K = 2 นั่นคือ ณ เวลาใด ๆ ก็ตาม นั่งร้านของลีโอนาร์โดนั้นเก็บสีได้ 2 สี ตามที่ได้กล่าวไว้ข้างต้น นั่งร้านจะเริ่มต้นโดยมีสีหมายเลข 0 และ 1 เราจะเขียนสิ่งที่เก็บบนนั่งร้านเป็น [0, 1] วิธีการหนึ่งที่ผู้ช่วยจะสามารถจัดการกับการร้องขอสีเป็นดังต่อไปนี้

- สีที่ถูกร้องขอสีแรก (หมายเลข 2) ไม่อยู่บนนั่งร้าน ผู้ช่วยเอาสีหมายเลข 2 ไปไว้บนนั่งร้านและตัดสินใจหยิบสีหมายเลข 1 ออกมาจากนั่งร้าน สถานการณ์ของนั่งร้านในขณะนี้เป็น [0, 2]
- สีถัดไป (หมายเลข 0) อยู่บนนั่งร้านอยู่แล้ว ดังนั้นผู้ช่วยสามารถพักได้
- สำหรับการร้องขอสีครั้งที่สาม (หมายเลข 3) ผู้ช่วยหยิบสีหมายเลข 0 ออก ทำให้สีบนนั่งร้านเป็น [3, 2]
- ในที่สุด สีที่ถูกกร้องขอสีสุดท้าย (หมายเลข 0) จะต้องถูกนำจากชั้นเก็บสีไปไว้บนนั่งร้าน ผู้ช่วยตัดสินใจเอาสีหมายเลข 2 ออกไปเก็บ สีบนนั่งร้านจะกลายเป็น [3, 0]

สังเกตว่า จากตัวอย่างข้างต้น ผู้ช่วยไม่ได้ทำตาม กลยุทธเหมาะสุดของลีโอนาร์โด ถ้าเป็นกลยุทธเหมาะสุดแล้ว ในขั้นที่สามจะต้องเลือกสีหมายเลข 2 ออกจากนั่งร้าน ด้วยวิธีการนี้ ผู้ช่วยจะสามารถพักได้อีกครั้งในขั้นตอนสุดท้าย

กลยุทธของผู้ช่วย เมื่อเขามีความทรงจำจำกัด

ในตอนเช้า ผู้ช่วยจะขอให้ลีโอนาร์โดเขียน C ลงบนแผ่นกระดาษ ซึ่งทำให้เขาสามารถหาและทำตามกลยุทธเหมาะสุดได้ อย่างไรก็ตามลีโอนาร์โดต้องการเก็บเทคนิคของเขาไว้เป็นความลับ ดังนั้น ลีโอนาร์โดจึงไม่ยอมให้ผู้ช่วยเอากระดาษแผ่นนั้นไป สิ่งเดียวที่เขาอนุญาตให้ผู้ช่วยทำ คือ ให้อ่าน C และพยายามจำให้ได้

โชคร้ายที่ความจำของผู้ช่วยนั้นแย่มาก เขาสามารถจำได้แค่ไม่เกิน M บิต ซึ่งโดยทั่วไปแล้วจะทำให้เขาไม่สามารถสร้างลำดับ C ทั้งก้อนขึ้นมาใหม่ ดังนั้นผู้ช่วยต้องแก้ปัญหาด้วยวิธีการชาญฉลาดบางวิธีที่คำนวณลำดับของบิตที่เขาจะท่องจำ เราจะเรียกลำดับนี้ว่า ลำดับแนะนำ เขียนแทนด้วย A

ตัวอย่างที่ 2

ในตอนเช้าผู้ช่วยจะสามารถนำกระดาษของลีโอนาร์โดมาอ่านลำดับ C และสร้างตัวเลือกที่จำเป็นทั้งหมดทางหนึ่งที่ผู้ช่วยสามารถเลือกทำได้ คือ ตรวจสอบสถานะของนั่งร้านหลังการร้องขอในแต่ละครั้ง ตัวอย่างเช่น เมื่อใช้วิธีการ (ที่เป็นวิธีการเกือบเหมาะสุด) ในตัวอย่างที่ 1 ลำดับของสถานะบนนั่งร้านจะเป็น [0, 2], [0, 2], [3, 2], [3, 0] (เขารู้ว่าสถานะเริ่มต้นของนั่งร้านคือ [0, 1])

ณ เวลานี้ สมมติว่า เรามี M = 16 ดังนั้น ผู้ช่วยจะสามารถจำข้อมูลได้ 16 บิต และเนื่องจาก N = 4 เราจะสามารถจำสีแต่ละสีโดยใช้ 2 บิต ดังนั้นการใช้บิต 16 บิตมาเก็บลำดับของสถานะบนนั่งร้านตามด้านบนนี้จึงเพียงพอ ทำให้ผู้ช่วยคำนวณลำดับแนะนำดังต่อไปนี้ A = (0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0).

หลังจากนั้นแล้ว ในระหว่างวันผู้ช่วยจะสามารถถอดรหัสลำดับแนะนำนี้ และใช้เป็นการเลือกของเขา

(แน่นอนว่าด้วย M = 16 ผู้ช่วยจะสามารถเลือกจำลำดับ C ทั้งก้อนได้เช่นกัน โดยใช้จำนวนบิต 8 บิตเท่านั้นจาก 16 ที่ใช้ได้ ในตัวอย่างนี้เป็นการแสดงให้เห็นว่า เขาอาจมีทางเลือกอื่นที่ทำได้อีกโดยที่ไม่ต้องระบุคำตอบโดยตรง)

ปัญหา

คุณต้องเขียน โปรแกรมสองอันแยกกัน ด้วยภาษาเดียวกัน โปรแกรมทั้งสองอันนี้จะถูกประมวลผลตามลำดับ โดยไม่สามารถรับส่งข้อมูลให้กันได้ในขณะที่ประมวลผลอยู่

โปรแกรมแรกถูกใช้โดยผู้ช่วยในตอนเช้า โปรแกรมนี้รับข้อมูลเป็นลำดับ C และต้องคำนวณลำดับแนะนำ A

โปรแกรมที่สองจะเป็นโปรแกรมที่ผู้ช่วยจะนำไปใช้ระหว่างวัน โปรแกรมนี้จะรับลำดับแนะนำ A แล้วโปรแกรมต้องประมวลผลลำดับ C ของการร้องขอจากลีโอนาร์โด ให้สังเกตว่า ลำดับ C จะถูกส่งให้โปรแกรมครั้งละหนึ่งการร้องขอ และการร้องขอแต่ละอันจะต้องถูกประมวลผลก่อนจะรับการร้องขออันถัดไปเข้ามา

หรือจะกล่าวให้ชัดขึ้นว่า ในโปรแกรมแรก คุณต้องเขียนโปรแกรมย่อยหนึ่งอันคือ `ComputeAdvice`(C, N, K, M) ซึ่งรับอินพุตเป็นอาเรย์ C ของจำนวนเต็ม N ตัว (แต่ละตัวมีค่า 0, …, N - 1) K เป็นจำนวนสีบนนั่งร้าน และ M เป็นจำนวนบิตที่ใช้ได้สำหรับคำแนะนำ โปรแกรมนี้ต้องคำนวณลำดับแนะนำ A ที่ประกอบด้วยบิตไม่เกิน M บิต โปรแกรมต้องส่งลำดับ A กับระบบแต่ละบิตเรียงตามลำดับ โดยเรียกใช้โปรแกรมย่อยต่อไปนี้

- `WriteAdvice`(B) — เอาบิต B ต่อท้ายลำดับแนะนำล่าสุด A (คุณสามารถเรียกฟังก์ชันนี้ได้ไม่เกิน M ครั้ง)

ในโปรแกรมที่สอง คุณต้องเขียนโปรแกรมย่อยหนึ่งอัน `Assist`(A, N, K, R) อินพุตของโปรแกรมย่อยอันนี้คือ ลำดับแนะนำ A ตัวเลขจำนวนเต็ม N และ K ตามนิยามข้างต้น และความยาว R มีหน่วยเป็นบิตของลำดับ A ($R \leq M$) โปรแกรมย่อยนี้ควร execute วิธีการที่คุณเสนอให้กับผู้ช่วย โดยใช้โปรแกรมย่อยที่เตรียมไว้ให้คุณ ดังนี้

- `GetRequest`() — ส่งกลับ สีถัดไป ที่ลีโอนาร์โดขอมา (ไม่มีการเปิดเผยรายละเอียดเกี่ยวกับการขอในอนาคต)
- `PutBack`(T) — ใส่ค่าสี T จากนั่งร้านกลับไปยังชั้นเก็บสี คุณสามารถเรียกฟังก์ชันนี้โดยที่ T จะต้องเป็นสีที่อยู่บนนั่งร้าน ณ เวลานั้นเท่านั้น

เมื่อถูก execute โปรแกรมย่อย `Assist` ของคุณต้องเรียก `GetRequest` N ครั้งพอดี ในแต่ละครั้งจะรับการร้องขอจากลีโอนาร์โดหนึ่งครั้งตามลำดับ หลังจากการเรียก `GetRequest` แต่ละครั้ง ถ้า สีที่คืนกลับมา ไม่อยู่ในนั่งร้าน คุณ ต้อง เรียก `PutBack`(T) ด้วย เมื่อ T เป็นการเลือกของคุณ ถ้าไม่เช่นนั้น คุณ ต้องไม่ เรียก `PutBack` การไม่กระทำตามนี้ จะถือเป็นข้อผิดพลาดซึ่งจะนำไปสู่การจบการทำงานของโปรแกรมคุณ โปรดระลึกว่าเมื่อเริ่มต้น นั่งร้านมีสีตั้งแต่ 0 ถึง K-1 (รวมหัวท้าย)

กรณีทดสอบที่ถือว่าถูกต้อง ถ้าโปรแกรมย่อยของคุณทั้งสองโปรแกรมเป็นไปตามเงื่อนไขที่กำหนดไว้ทั้งหมด และจำนวนการเรียก `PutBack` ทั้งหมด ต้อง เท่ากับ จำนวนที่คำนวณได้จากกลยุทธเหมาะสุดของลีโอนาร์โด ให้สังเกตว่า ถ้ามีวิธีการหลายอันที่ให้จำนวนการเรียก `PutBack` เท่ากัน โปรแกรมของคุณจะสามารถเลือกทำอันใดก็ได้ (นั่นคือ ไม่จำเป็นต้องเป็นไปตามกลยุทธเหมาะสุดของลีโอนาร์โดเสมอไป ถ้ามีกลยุทธอื่นที่ดีเท่า ๆ กัน)

ตัวอย่างที่ 3

ต่อเนื่องจากตัวอย่างที่ 2, สมมติว่าใน `ComputeAdvice` คุณคำนวณ A = (0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0) คุณอาจจะใช้ลำดับการเรียกโปรแกรมย่อยดังต่อไปนี้ เพื่อที่จะรับส่งข้อมูลกับระบบ `WriteAdvice(0)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(0)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(0)`

โปรแกรมย่อยที่สองของคุณ `Assist` จะถูก execute จะได้รับลำดับ A ข้างต้น และค่า N = 4, K = 2 และ R = 16 โปรแกรมย่อย `Assist` จะต้องเรียกใช้ `GetRequest` 4 ครั้งพอดี (N = 4) และเมื่อทำงานตามร้องขอไปบ้างแล้ว `Assist` จะต้องเรียกใช้ `PutBack(T)` โดยต้องกำหนดค่า T ให้เหมาะสม

ตารางด้านล่างแสดงลำดับการเรียกใช้ฟังก์ชันที่เป็นไปตาม ทางเลือกที่ เกือบเหมาะสุด ตามตัวอย่างที่ 1 เครื่องหมาย "-" (hyphen) แสดงถึงการไม่ได้เรียกใช้ `PutBack`

`GetRequest`()	Action
2	`PutBack`(1)
0	-
3	`PutBack`(0)
0	`PutBack`(2)

งานย่อยที่ 1 [8 คะแนน]

- $N \leq 5,000$
- คุณสามารถใช้ได้มากที่สุด M = 65,000 บิต

งานย่อยที่ 2 [9 คะแนน]

- $N \leq 100,000$
- คุณสามารถใช้ได้มากที่สุด M = 2,000,000 บิต

งานย่อยที่ 3 [9 คะแนน]

- $N \leq 100,000$
- $K \leq 25,000$
- คุณสามารถใช้ได้มากที่สุด M = 1,500,000 บิต

งานย่อยที่ 4 [35 คะแนน]

- $N \leq 5\ 000.$
- คุณสามารถใช้ได้มากที่สุด M = 10,000 บิต

งานย่อยที่ 5 [ได้มากถึง 39 คะแนน]

- $N \leq 100,000$
- $K \leq 25,000$
- คุณสามารถใช้ได้มากที่สุด M = 1,800,000 บิต

คะแนนของงานย่อยนี้ขึ้นอยู่กับความยาว R ของคำแนะนำที่โปรแกรมของคุณรับและส่ง นั่นคือ ถ้า R_{max} เป็นค่าสูงสุด (ในทุกชุดทดสอบทุกชุด) ของความยาวของลำดับแนะนำที่สร้างโดยโปรแกรมย่อย

ComputeAdvice คะแนนของคุณจะเป็น:

- 39 คะแนน ถ้า $R_{max} \leq 200,000$
- 39 $(1,800,000 - R_{max}) / 1,600,000$ คะแนน ถ้า $200,000 < R_{max} < 1,800,000$;
- 0 คะแนน ถ้า $R_{max} \geq 1,800,000$

รายละเอียดของการโปรแกรม

คุณควรส่งสองโปรแกรม ที่เขียนด้วย ภาษาโปรแกรมเดียวกัน

แฟ้มแรกมีชื่อว่า advisor.c, advisor.cpp หรือ advisor.pas ในแฟ้มนี้จะต้องเขียนโปรแกรมย่อย ComputeAdvice ตามที่อธิบายไว้ข้างต้น และสามารถเรียกโปรแกรมย่อย WriteAdvice แฟ้มที่สองมีชื่อว่า assistant.c, assistant.cpp หรือ assistant.pas. แฟ้มนี้จะต้องเขียนโปรแกรมย่อย Assist ตามที่อธิบายไว้ข้างต้น และสามารถเรียกโปรแกรมย่อย GetRequest และ PutBack.

รูปแบบของโปรแกรมย่อยเป็นดังต่อไปนี้

ภาษา C/C++

```
void ComputeAdvice(int *C, int N, int K, int M);
void WriteAdvice(unsigned char a);



void Assist(unsigned char *A, int N, int K, int R);
void PutBack(int T);
int GetRequest();
```

ภาษา Pascal

```
procedure ComputeAdvice(var C : array of LongInt; N, K, M : LongInt);
procedure WriteAdvice(a : Byte);



procedure Assist(var A : array of Byte; N, K, R : LongInt);
procedure PutBack(T : LongInt);
function GetRequest : LongInt;
```

โปรแกรมย่อยเหล่านี้ ต้องทำงานตามที่อธิบายไว้ด้านบน โดยคุณสามารถเขียนโปรแกรมย่อยอื่น ๆ เพื่อใช้ภายใน สำหรับโปรแกรมภาษา C/C++ โปรแกรมย่อยภายในของคุณควรถูก declare เป็น static เพื่อให้ grader ตัวอย่าง link โปรแกรมเข้าด้วยกัน หรือ อาจหลีกเลี่ยงที่จะมีโปรแกรมย่อยสองตัวนี้ (หนึ่งโปรแกรมย่อยในแต่ละไฟล์) ไม่ให้มีชื่อซ้ำกัน โปรแกรมของคุณจะต้องไม่ยุ่งเกี่ยวกับ standard input/output หรือแฟ้มใด ๆ

ในช่วงที่คุณเขียนโปรแกรมอยู่นั้น คุณต้องระมัดระวังและทำตามคำสั่งด้านล่าง (เทมเพลตที่เตรียมไว้ให้นั้นทำตามคำสั่งด้านล่างเรียบร้อยแล้ว)

ภาษา C/C++

ในส่วนแรกของคำตอบของคุณ คุณต้อง include ไฟล์ advisor.h และ assistant.h ในตัวผู้ให้คำแนะนำและตัวผู้ช่วยตามลำดับ ซึ่งทำได้โดยการใส่บรรทัด

```
#include "advisor.h"
```

หรือ

```
#include "assistant.h"
```

เราจะเตรียมไฟล์สองอัน คือ advisor.h และ assistant.h ไว้ให้ใน directory ที่อยู่ในเครื่องของคุณ และอยู่ใน Web interface ของการแข่งขัน นอกจากนี้ยังมี code และสคริปต์ที่ใช้คอมไพล์และทดสอบคำตอบของคุณ โดยเฉพาะอย่างยิ่ง หลังจากก็อปปี้คำตอบของคุณลงใน directory ด้วยสคริปต์เหล่านี้ คุณจะต้อง run คำสั่ง compile_c.sh หรือ compile_cpp.sh (ขึ้นอยู่กับภาษาที่คุณใช้)

ภาษา Pascal

คุณจำเป็นต้องเรียกใช้ units advisorlib และ assistantlib ตามลำดับ ในตัวผู้ให้คำแนะนำและตัวผู้ช่วยตามลำดับ โดยใส่ส่วนโปรแกรมต่อไปนี้ในโปรแกรมของคุณ

```
uses advisorlib;
```

หรือ

```
uses assistantlib;
```

เราจะเตรียมไฟล์สองอัน คือ advisorlib.pas และ assistantlib.pasไว้ให้ใน directory ที่อยู่ในเครื่องของคุณ และอยู่ใน Web interface ของการแข่งขัน นอกจากนี้ยังมี code และสคริปต์ที่ใช้คอมไพล์และทดสอบคำตอบของคุณ โดยเฉพาะอย่างยิ่ง หลังจากก็อปปี้คำตอบของคุณลงใน directory ด้วยสคริปต์เหล่านี้ คุณจะต้อง run คำสั่ง compile_pas.sh

grader ตัวอย่าง

grader ตัวอย่างจะรับข้อมูลนำเข้าตามรูปแบบต่อไปนี้:

- บรรทัดที่ 1: N, K, M;
- บรรทัดที่ 2, …, N + 1: C[i].

grader ตัวอย่างจะเริ่มต้นด้วยการประมวลผลฟังก์ชัน ComputeAdvice ซึ่งจะสร้างแฟ้ม advice.txt โดยในแฟ้มนี้จะมีบิตของลำดับคำแนะนำ คั่นด้วยช่องว่างและจบด้วย 2

หลังจากนั้น grader ตัวอย่างจะประมวลผลโปรแกรมย่อย Assist ของคุณ มันจะสร้างข้อมูลส่งออกโดยที่แต่ละบรรทัดจะเป็นในรูปแบบ "R [number]" หรือ "P [number]" โดยแบบแรกจะเป็นการเรียกไปยัง GetRequest() และคอยรับค่าตอบกลับ และแบบที่สองจะเป็นการเรียกไปยัง PutBack() และสีที่เลือกจะถูกเอาไปเก็บ ข้อมูลส่งออกจะจบด้วยบรรทัดในรูปแบบ "E"

โปรดเข้าใจว่าการใช้ official grader ตรวจโปรแกรมของคุณ จะมีเวลาในการประมวลผลที่แตกต่างเล็กน้อยกับที่ทำงานบนคอมพิวเตอร์ของคุณ ความแตกต่างนี้ไม่น่ามีนัยสำคัญ อย่างไรก็ตาม เราแนะนำให้คุณใช้

test interface เพื่อทดสอบว่าโปรแกรมของคุณโปรแกรมของคุณจะทำงานไม่เกินข้อจำกัดของเวลา

ข้อจำกัดของเวลาและหน่วยความจำ

- ข้อจำกัดของเวลา: 7 วินาที
- ข้อจำกัดของหน่วยความจำ: 256 MiB