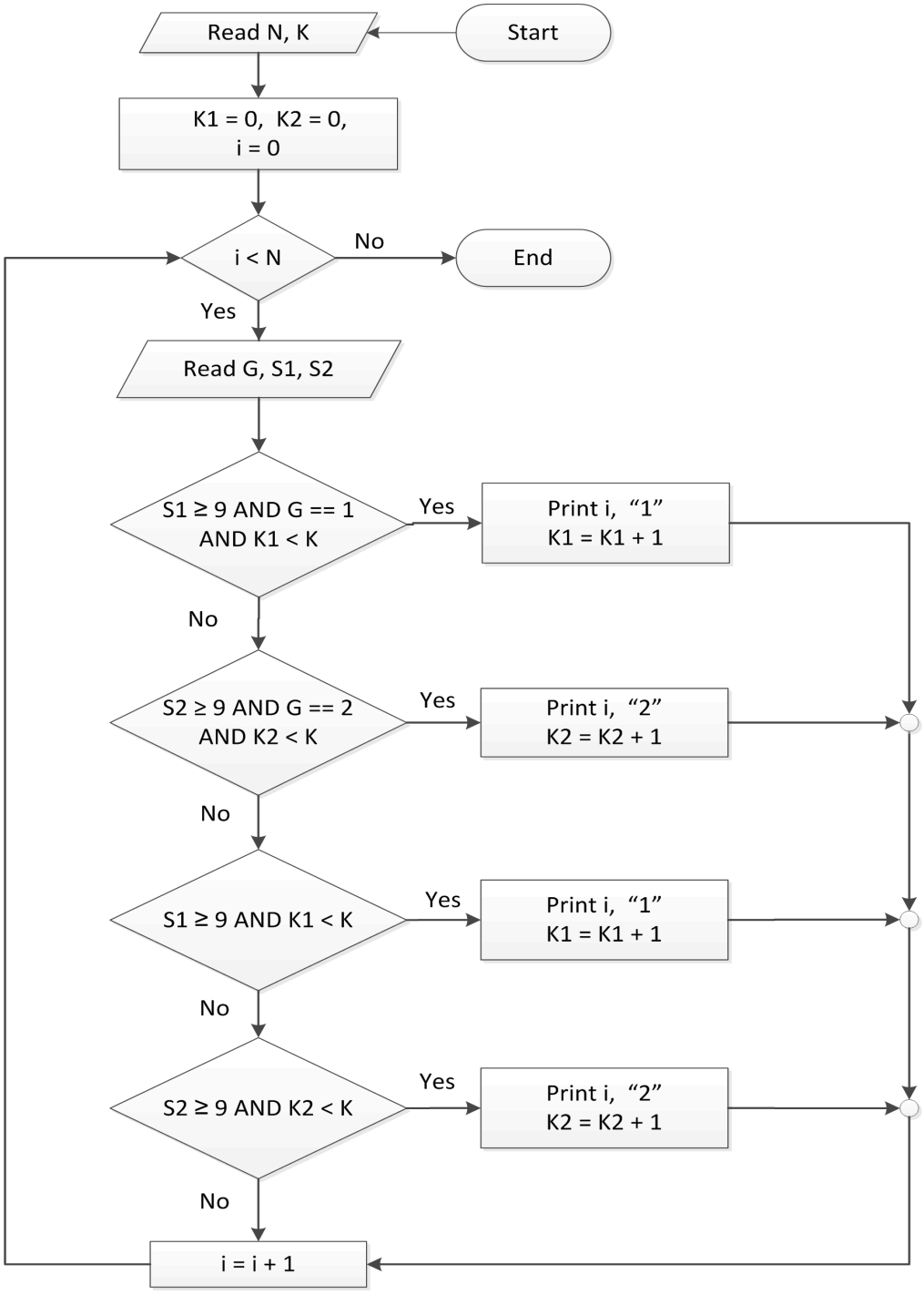
ข้อ 13 “คัดเลือกนักร้อง”

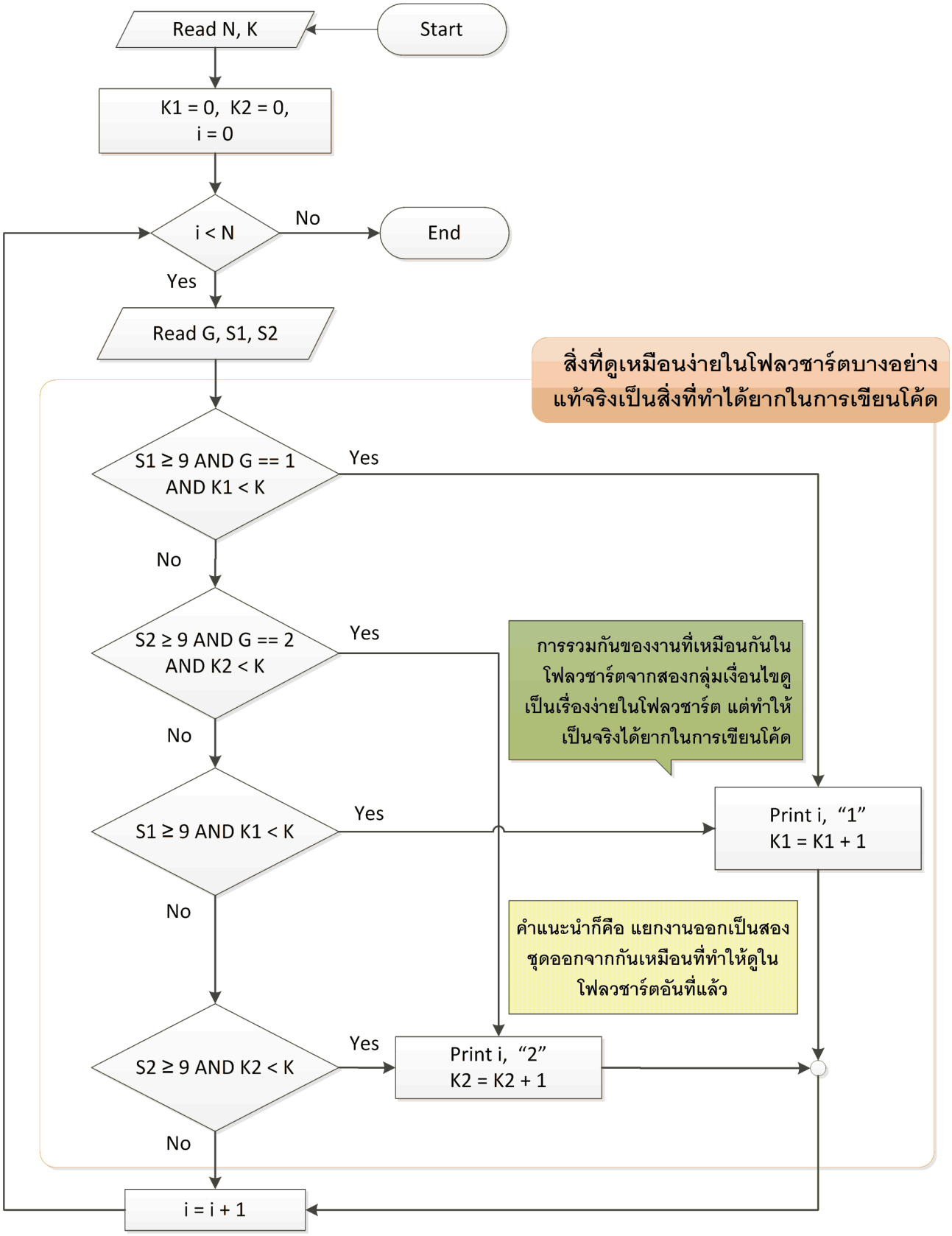
สำหรับคนที่ต้องการเขียนโฟลวชาร์ตด้วย สามารถดูโฟลวชาร์ตข้างล่างนี้เป็นแนวทางได้ เช่นเดียวกับข้อที่แล้ว ผลลัพธ์ทุกอย่างถูกพิมพ์ออกมาในลูปหมดแล้ว ดังนั้นเมื่อออกจากลูป โปรแกรมจะจบการทำงานทันที และสิ่งที่เราควรจะสังเกตเพิ่มเติมก็คือว่าในลูปบางลูปจะไม่มีการพิมพ์ค่าใด ๆ ออกมา แต่เหมือนจะผ่านไปอย่างเงียบ ๆ โดยไม่มีอะไรเกิดขึ้น ซึ่งเกิดขึ้นในกรณีที่ผู้เข้าประกวดตกรอบ เนื่องจากคะแนนไม่ถึงหรือกรรมการมีผู้เข้าประกวดในการดูแลครบตามจำนวนแล้ว



จากโฟลวชาร์ตที่แสดงไว้ เราสามารถนำมาเขียนเป็นซูโดโค้ดที่เทียบเท่ากันได้เป็น

```
START
  READ N, K
  K1 = 0
  K2 = 0
  i = 0
  WHILE i < 0 DO
    READ G, S1, S2
    IF S1 >= 9 AND G == 1 AND K1 < K THEN
      PRINT i, “1”
      K1 = K1 + 1
    ELSE IF S2 >= 9 AND G == 2 AND K2 < K THEN
      PRINT i, “2”
      K2 = K2 + 1
    ELSE IF S1 >= 9 AND K1 < K THEN
      PRINT i, “1”
      K1 = K1 + 1
    ELSE IF S2 >= 9 AND K2 < K THEN
      PRINT i, “2”
      K2 = K2 + 1
    END IF
    i = i + 1
  END WHILE
END
```

แต่ปัญหานี้ ที่จริงมีเรื่องลึกลับชวนฉงนอยู่อย่างหนึ่ง นั่นก็คือว่าเราเห็นคำสั่ง ที่ซ้ำกัน เช่นมี Print i, “1” สองครั้ง และก็มีคำสั่ง Print i, “2” ซ้ำกันสองครั้งเช่นกัน หลายคนอาจจะคิดว่าแบบนี้ควรจะจับรวมกันเป็นแบบโฟลวชาร์ตข้างล่าง



อย่างไรก็ตามเรื่องเศร้าก็คือว่า ในโฟลวชาร์ต เราสามารถลากเส้นให้ข้ามหรือย้อนไปมาได้โดยง่าย แต่ในโค้ดเราอาจจะ

ไม่สามารถทำเช่นนั้นได้โดยสะดวกในบางกรณี เช่นในตัวอย่างนี้ถึงงานจะเหมือนกัน แต่ถ้าเราคิดว่ามันเป็นชิ้นเดียวกันจริง ๆ ก็หมายความว่า เราโยกลำดับการทำงานใน IF อันหนึ่งไปรวมใส่ไว้ใน IF อีกอันหนึ่ง แม้ในภาษาซีสามารถทำได้ แต่ความสามารถนั้นทำให้โค้ดดูซับซ้อนผิดปรกติ และความสามารถนี้ถูกเอาออกในภาษายุคต่อมาอย่างภาษาจาวาเพื่อป้องกันไม่ให้เกิดความยุ่งเหยิงในโค้ด

ทางออกอีกทางหนึ่งที่ทำให้โค้ดไม่ยุ่งเหยิงและเป็นที่ยอมรับกันอย่างแพร่หลายก็คือการแยกงานที่ใช่ร่วมกันนั้นออกเป็นฟังก์ชัน และทำการเรียกใช้ฟังก์ชันนั้นในจุดที่เราต้องการ (การสร้างฟังก์ชันสำหรับงานที่เหมือนกันแบบนี้เป็นที่นิยมมาก และเราจะเรียนเรื่องฟังก์ชันในครึ่งหลังของภาคการศึกษา)

แน่นอนว่า หลายคนยังไม่หยุดสงสัย เค้าคิดว่าไม่ว่าจะอย่างไร เค้าต้องเอาการทำงานที่ดูซ้ำกันแบบนี้มารวมกันให้ได้ในรูปแบบที่ภาษาสมัยใหม่ยอมรับ ซึ่งความพยายามดังกล่าวแท้จริงก็สามารถทได้อย่างถูกสุขลักษณะหากเราจะปรับเงื่อนไขในการพิจารณาให้ซับซ้อนขึ้นเล็กน้อย

ในปัญหานี้ เราต้องกลับไปที่หลักการพิจารณาผู้เข้าประกวด หากเราจะรวมให้เหลือ Print i, 1 เพียงตำแหน่งเดียว เราก็จะต้องคิดหาเหตุผลที่เพียงพอที่จะตอบว่านักร้องจะได้อยู่กับกรรมการคนที่หนึ่งโดยครอบคลุมทุกกรณีที่เป็นไปได้ และสำหรับกรรมการคนที่สองก็ต้องคิดในลักษณะเดียวกัน ทำให้เราได้ซูโดโค้ดออกมาเป็นแบบข้างล่าง [เราใช้ซูโดโค้ดตรงนี้เพื่อแสดงให้เห็นว่า เงื่อนไขที่รอบครอบจะทำให้เรายุบรวมงานได้ตามจุดประสงค์ที่วางไว้ในตอนแรก]

```
START
  READ N, K
  K1 = 0
  K2 = 0
  i = 0
  WHILE i < 0 DO
    READ G, S1, S2
    IF (S1 >= 9 AND G == 1 AND K1 < K) OR
       (S1 >= 9 AND K1 < K AND K2 == K) OR
       (S1 >= 9 AND K1 < K AND S2 < 9) THEN
      PRINT i, "1"
      K1 = K1 + 1
    ELSE IF (S2 >= 9 AND G == 2 AND K2 < K) OR
            (S2 >= 9 AND K2 < K AND K1 == K) OR
            (S2 >= 9 AND K2 < K AND S1 < 9) THEN
      PRINT i, "2"
      K2 = K2 + 1
    END IF
    i = i + 1
  END WHILE
END
```

จุดสำคัญก็คือว่า เราต้องตรวจเพิ่มเติมว่ากรรมการอีกคนไม่มีพื้นที่แล้ว หรือกรรมการอีกคนให้คะแนนน้อย ดังนั้นเราจึงสามารถตัดสินได้ทันทีว่าผู้เข้าประกวดจะอยู่กับกรรมการคนใด ยิ่งไปกว่านั้น หากเราใช้ความรู้ทางตรรกศาสตร์มาพิจารณาเงื่อนไข เราก็จะพบว่าเราสามารถย่อเงื่อนไขกลุ่มแรกได้เป็น

```
(S1 >= 9 AND K1 < K) AND (G == 1 OR K2 == K OR S2 < 9)
```

นั่นคือถ้ากรรมาการคนแรกให้คะแนนมากและยังมีพื้นที่เหลือ เราต้องการเงื่อนไขเพิ่มอีกเล็กนอ้ยก็จะเพียงพอที่จะตัดสินได้อย่างถูกต้องทันที่ว่าผู้เข้าแข่งขันจะอยู่กับกรรมการคนแรกหรือไม่ ซึ่งเงื่อนไขดังกล่าวก็คือว่า ถ้าหากผู้เข้าประกวดเป็นผู้ชาย หรือ กรรมการอีกคนไม่เหลือพื้นที่แล้ว หรือกรรมการอีกคนให้คะแนนน้อยสำหรับการย่อเงื่อนไขกลุ่มที่สองก็ทำได้ในลักษณะเดียวกัน

อย่างไรก็ตาม ในกรณีของกรรมการคนที่สอง หากเราใช้ประโยชน์จากการทำงานของ ELSE IF เราจะพบว่าถ้าหากจะมาถึงเงื่อนไข ELSE IF ได้ เรายอ่มทราบว่านักร้องคนนี้จะไม่อยู่กับกรรมการคนที่ 1 แล้วแน่นอน เราจึงสามารถลดทอนเงื่อนไขให้เหลือสั้น ๆ เพียง (S2 >=9 AND K2 < K) ก็เพียงพอแล้ว และเมื่อรวมการปรับปรุงต่าง ๆ ไว้ด้วยกัน เราจะได้โค้ดที่กะทัดรัดมากขึ้นเป็น

```
START
  READ N, K
  K1 = 0
  K2 = 0
  i = 0
  WHILE i < 0 DO
    READ G, S1, S2
    IF (S1 >= 9 AND K1 < K) AND (G == 1 OR K2 == K OR S2 < 9) THEN
      PRINT i, “1”
      K1 = K1 + 1
    ELSE IF (S2 >= 9 AND K2 < K) THEN
      PRINT i, “2”
      K2 = K2 + 1
    END IF
    i = i + 1
  END WHILE
END
```