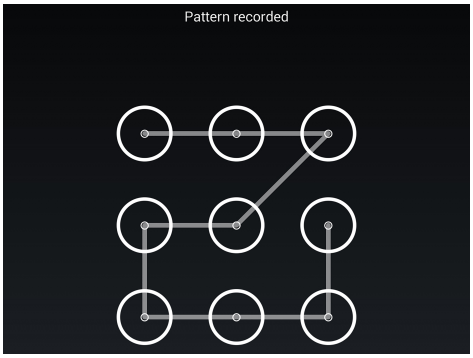
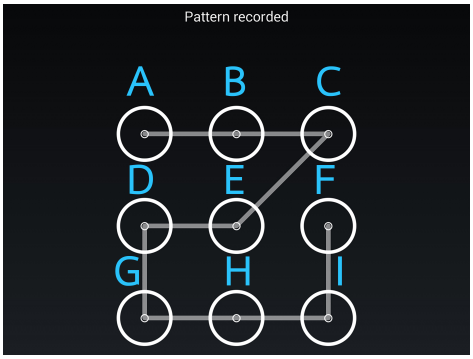
แอนดรอยด์ (Andriod)

โทรศัพท์ Andriod จะมีวิธีการปลดล็อกด้วยรหัสแบบหนึ่ง คือ Pattern Password ซึ่งเป็นเทคโนโลยีที่ล้ำมาก ๆ เพราะแม้แต่ iOS ก็คาดคิดไม่ถึง

นักศึกษาเป็นผู้ใช้โทรศัพท์ Android และต้องการที่จะมีรหัสที่ปลอดภัยที่สุด ดังนั้นนักศึกษาเลยใส่รหัสโดยมีจำนวนจุด Pattern มากที่สุดที่เป็นไปได้ ดังตัวอย่างรูปด้านล่างนี้



หากเราให้แต่ละจุดแทนด้วยเครื่องหมาย



เราจะได้ว่า นักศึกษามีลำดับของรูปแบบนี้คือ **ABCEDGHIF** หรือในอีกกรณีอาจจะเป็น **FIHGDECBA** *(เพราะว่าเราไม่ได้เห็นทิศทางของรูป)* เพื่อให้เข้าใจตรงกัน เราจะอนุมานให้เป็นรูปแบบ **ABCEDGHIF**

ในเช้าวันหนึ่ง นักศึกษาตื่นมาแล้วปรากฏว่า **ลืม** รหัสที่ตัวเองได้ตั้งไว้ว่ามันคือรหัสอะไรกันแน่ ! นักศึกษารู้แค่ว่ารูปแบบที่นักศึกษาตั้ง ต้องใช้ทุกจุด (ในกรณีด้านบนคือทั้งหมด 9 จุด) ดังนั้นนักศึกษาเลยลองวาดจุดทุกรูปแบบที่เป็นไปได้ (สมมติว่าไม่มี Feature ในการล็อกโทรศัพท์หลังใส่รหัสผิดไปหลาย ๆ ครั้ง) นักศึกษาเลยสามารถใส่รหัสในทุกรูปแบบที่เป็นไปได้ เพื่อหารหัสที่ถูกต้อง

โชคดีที่ว่านักศึกษาเพิ่งเรียนการ Generate Permutation จากอ.วีเมื่อสักครู่นี้ เพื่อความเป็นระเบียบ เราจะเรียงรูปแบบรหัสที่จะลองตาม Lexicographic คือเรียงตามพจนานุกรม (จะได้ไม่ใส่รหัสแบบมั่ว ๆ สะเปะสะปะไปเรื่อย)

ทีนี้นักศึกษาเลยเริ่มใส่รหัสจาก **ABCDEFGHI** ซึ่งจะเป็นลำดับที่ 1 ต่อจากนั้นตามลำดับพจนานุกรมจะเป็น **ABCDEFGIH** เป็นลำดับที่ 2 ใส่ไปเรื่อย ๆ จนลองใส่ **ABCEDGHIF** เป็นลำดับที่ 130 จึงได้รหัสที่ถูกต้อง ดังนั้นจำนวนครั้งในการลองใส่คือ 130 ครั้ง

ทีนี้นักศึกษาเลยอยากลองให้คุณเขียนโปรแกรมหาว่า หากในอนาคตตั้งรหัสเป็นอย่างอื่นแล้วดันลืมอีก จะต้องลองใส่รหัสกี่ครั้งตามลำดับพจนานุกรม ถึงจะได้รหัสที่ถูกต้อง ตัวอย่างเช่น หากเราตั้งรหัสเป็น **IHGFEDCBA** เราจะต้องลองใส่ทั้งหมด 362880 ครั้งจึงจะได้รหัสที่ถูกต้อง แต่หากเราตั้งรหัสง่าย ๆ เช่น **ABCDEFGHI** ใส่แค่ 1 ครั้งก็ได้รหัสแล้ว

งานของนักศึกษา

จงหาว่า หากนักศึกษามีจำนวนจุด Pattern ทั้งหมด n จุด และมีชุดรหัสทั้งหมด m รหัส แต่ละรหัสต้องผ่านการลองใส่กี่ครั้งตามลำดับพจนานุกรม จึงจะสามารปลดล็อกโทรศัพท์ได้ กำหนดให้รหัสต้องมีทั้งหมด n ตัวเท่านั้น และ**ห้าม**นักศึกษาใช้ฟังก์ชันอย่าง `next_permutation()` โดยเด็ดขาด

ข้อมูลนำเข้า (Input)

บรรทัดที่ 1	จำนวนเต็ม n แทนจำนวนจุดใน Pattern และ m แทนจำนวนคำถามที่จะถาม โดยที่ $3 \leq n \leq 11$ และ $1 \leq m \leq 10$
บรรทัดที่ 2 ถึง $m+1$	รูปแบบของรหัสที่นักศึกษาอยากรู้ โดยประกอบไปด้วยตัวอักษรทั้งหมด n ตัว

ข้อมูลส่งออก (Output)

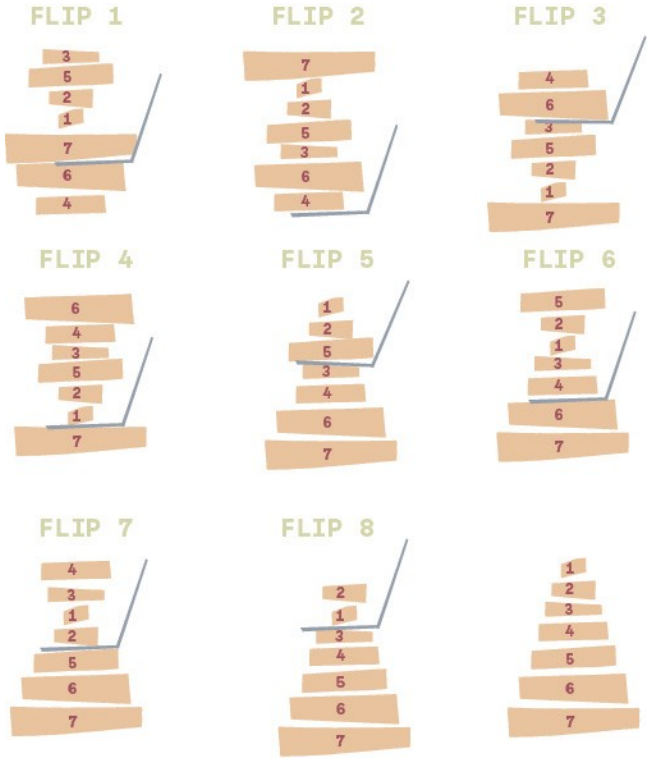
บรรทัดที่ 1 ถึง m	จำนวนครั้งที่ต้องใส่รหัสดังกล่าวตามลำดับพจนานุกรม จึงจะสามารถปลดล็อกโทรศัพท์ได้

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
9 3 ABCEDGHIF IHGFEDCBA ABCDEFGHI	130 362880 1
5 5 ABCED EDCAB BCEAD CDAEB ADECB	2 119 35 62 18

Pancake Sort

Pancake sort เป็น algorithm ที่ใช้สำหรับการจัดเรียงซึ่งมีรูปแบบมาจากพฤติกรรมการพลิก(flip)ของ pancake โดยใช้หลักการของการพลิกข้อมูลจากตำแหน่งแรกไปจนถึงตำแหน่งหนึ่ง เพื่อให้ได้ลำดับที่ต้องการ



โดยที่มาที่ไปของ algorithm นี้เกิดจากการจัดวาง pancake บนจาน โดยการวางนั้นจำเป็นต้องทำให้ชิ้นที่ใหญ่กว่าอยู่ด้านล่าง และชิ้นที่เล็กกว่าอยู่ด้านบน จะทำให้ได้การเรียงลำดับจากน้อยไปมาก

หลักการทำงานของ Pancake sorting

- หาแพนเค้กที่ใหญ่ที่สุดในส่วนที่เหลืออยู่: เริ่มต้นจากแพนเค้กทั้งกอง เลือกแพนเค้กที่ใหญ่ที่สุดจากกองที่ยังไม่ได้จัดเรียง (สมมติว่าเป็นส่วนย่อยของกองแพนเค้กทั้งหมดที่ยังเหลืออยู่)
- พลิกแพนเค้กใหญ่ที่สุดให้อยู่บนสุด: ถ้าแพนเค้กที่ใหญ่ที่สุดไม่ได้อยู่บนสุดของกอง ให้ทำการพลิกแพนเค้กจากตำแหน่งที่มันอยู่ไปจนถึงบนสุด เพื่อให้แพนเค้กใหญ่สุดมาอยู่บนสุด
- พลิกอีกครั้งเพื่อให้แพนเค้กใหญ่อยู่ในตำแหน่งที่ถูกต้อง: จากนั้นให้ทำการพลิกแพนเค้กจากบนสุดจนถึงตำแหน่งที่ต้องการ (ซึ่งก็คือตำแหน่งที่มันควรอยู่ในลำดับที่เรียงถูกต้อง) เพื่อให้แพนเค้กที่ใหญ่สุดลงไปอยู่ในตำแหน่งสุดท้าย
- ทำซ้ำ: จากนั้นทำซ้ำขั้นตอนเดียวกันสำหรับแพนเค้กที่เหลือ จนกระทั่งแพนเค้กทั้งหมดถูกจัดเรียงในลำดับที่ถูกต้อง

ก่อนที่จะเข้าสู่การเขียนโปรแกรม ให้นักศึกษาเติม Pseudocode ในช่องว่างของกระดาษในคาบเรียนให้เรียบร้อยก่อน

งานของนักศึกษา

ให้รับค่าตัวเลขจำนวนเต็ม n จำนวนจากผู้ใช้ จากนั้นทำการเรียงจำนวนที่รับเข้ามาโดยใช้ Pancake sorting algorithm และส่งออกเป็นลำดับของจำนนที่รับเข้ามา ที่ผ่านการเรียงลำดับจากน้อยไปมากเรียบร้อยแล้ว

ข้อมูลนำเข้า (Input)

บรรทัดที่ 1	จำนวนเต็ม n แสดงจำนวนของ Pancake
บรรทัดที่ 2	จำนวนเต็ม n จำนวน แสดงขนาดของ Pancake แต่ละชิ้น

ข้อมูลส่งออก (Output)

บรรทัดที่ 1	จำนวนเต็ม n จำนวนที่ผ่านการเรียงลำดับจากน้อยไปมาก

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
5 9 9 4 5 6	4 5 6 9 9
7 8 7 6 5 4 3 2	2 3 4 5 6 7 8

คำแนะนำ เพื่อไม่ให้ยุ่งยาก เรามีฟังก์ชัน `swap()` ในภาษา C++ ให้ใช้นะ อยู่ในไลบรารี่ `<algorithm>`

Quick Select

Quickselect ที่ใช้ Lomuto partition ทำงานโดยแบ่ง list เป็นสองส่วนตามแนวทางของ Lomuto partition scheme ซึ่งเป็นการเลือก pivot และแบ่งlistให้ pivot อยู่ในตำแหน่งที่ถูกต้องหลังจากการ partition แต่ละครั้ง

ขั้นตอนการทำงานของ Quickselect โดยใช้ Lomuto Partition:

1. เลือก pivot: ปกติแล้วเลือก pivot เป็นตัวสุดท้ายของ list
2. ทำ Lomuto Partition: เลื่อนค่าที่น้อยกว่าหรือเท่ากับ pivot ไปทางซ้าย และค่าที่มากกว่า pivot ไปทางขวา:
   - กำหนดตัวชี้ i ไว้ที่ตำแหน่งแรก
   - เปรียบเทียบทุกค่าสำหรับตำแหน่ง j กับ pivot ถ้าค่าที่ตำแหน่ง j น้อยกว่าหรือเท่ากับ pivot ให้สลับค่าที่ตำแหน่ง i กับ j แล้วเลื่อนตัวชี้ i ไปข้างหน้า
   - สุดท้าย สลับ pivot ไปที่ตำแหน่งที่เหมาะสม (ตำแหน่งของ i) ทำให้ pivot อยู่ในตำแหน่งที่ถูกต้อง
3. ตรวจสอบตำแหน่งของ pivot:
   - ถ้าตำแหน่งของ pivot ตรงกับค่า k ที่เราต้องการ ให้คืนค่า pivot นั้นเป็นคำตอบ
   - ถ้า pivot อยู่ในตำแหน่งมากกว่า k ให้ทำ Quickselect กับลิสต์ฝั่งซ้ายของ pivot
   - ถ้า pivot อยู่ในตำแหน่งน้อยกว่า k ให้ทำ Quickselect กับลิสต์ฝั่งขวาของ pivot
4. ทำซ้ำจนกว่าจะหาคำตอบได้: ทำขั้นตอนการเลือก pivot และแบ่งlistต่อไป จนกว่าจะพบค่าอันดับที่ต้องการ

งานของนักศึกษา

ให้รับค่าตัวเลขจำนวนเต็ม n จำนวนจากผู้ใช้ จากนั้นทำการหาจำนวนที่น้อยที่สุดลำดับที่ k จากชุดจำนวนเต็มที่รับเข้ามา และแสดงผล โดยที่ $k \leq n$

ข้อมูลนำเข้า (Input)

บรรทัดที่ 1	จำนวนเต็ม n แสดงถึงจำนวนของตัวเลขใน list
บรรทัดที่ 2	จำนวนเต็ม n แสดงสมาชิกในแต่ละตัว list
บรรทัดที่ 3	จำนวนเต็ม k แสดงตำแหน่งตัวเลขที่น้อยที่สุดลำดับที่ k

ข้อมูลส่งออก (Output)

บรรทัดที่ 1	จำนวนเต็มที่เป็นสมาชิกของ list ที่เข้ามาและเป็นจำนวนที่น้อยที่สุดลำดับที่ k

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
8 9 5 7 3 2 7 0 1 4	3
10 12 5 76 90 3 5 15 3 21 7 6	12
5 1 2 3 4 5 4	4