Vector Some Move
(1 s, 128 MB)

สำหรับโจทย์ข้อนี้จะมีความคล้ายคลึงกับข้อ Vector No Move ซึ่งได้นิยามไว้ดังนี้ **(หากรู้นิยามของ CP:vector_no_move อยู่แล้วสามารถข้ามไปหน้าถัดไปได้)**

ในโจทย์ข้อนี้ เราจะปรับเปลี่ยน CP::vector ใหม่ โดยไม่ให้มีการย้ายข้อมูล แต่จะใช้วิธีเก็บ Array ย่อย หลาย ๆ ตัวแทน และปรับเปลี่ยนการเพิ่มข้อมูลเมื่อ vector เต็ม เป็นการสร้าง Array ใหม่เท่ากับจำนวนช่องที่ต้องใช้เพิ่มขึ้นเท่านั้น และเก็บ array เก่าไว้ด้วย

ตัวอย่างเช่น หาก mCap = mSize = 4 และ ใช้ Array 1 ตัวในการเก็บข้อมูลเหล่านั้น และต้องการเพิ่มข้อมูลเข้าไป 2 ตัวแล้ว vector ใหม่นี้จะสร้าง array ใหม่เพียง 2 ช่อง ทำให้vector ใหม่นี้มี Array อยู่ 2 ตัว โดยที่ตัวแรกเก็บข้อมูล ช่องหมายเลข [0] ถึง [3] ของ vector และ array ตัวที่สอง เก็บช่องหมายเลข [4] ถึง [5] ของ vector

หากหลังจากนั้นมีการเพิ่มข้อมูลอีก 8 ช่อง vector ก็จะทำการสร้าง array ใหม่จำนวน 8 ช่อง ทำให้มี Array จำนวน 3 ตัว โดยที่สองตัวแรกไม่เปลี่ยนแปลง ส่วน array ตัวที่สามนั้นจะเก็บช่องหมายเลข [6] ถึง [13] ของ vector

ในโจทย์ข้อนี้จะมีไฟล์ของคลาส CP::vector_no_move ซึ่งเป็นคลาสของ vector แบบใหม่นี้เพื่อความสะดวก vector ใหม่นี้จะมีฟังก์ชันเพียงแค่ constructor, push_back, resize และ operator[] เท่านั้น **(ซึ่งได้ Implement มาเรียบร้อยแล้ว)** โดยยังไม่รองรับการใช้งาน iterator หรือฟังก์ชันอื่น ๆ ส่วนข้อมูลที่คลาสนี้เก็บ ได้แก่ mSize และ mCap นั้นการใช้งานเหมือน vector ทั่วไป แต่ mData จะถูกประกาศเป็น std::vector<std::vector<T>> mData แทน กล่าวคือ mData[i] จะเป็น std::vector<T> ซึ่งแทน array แต่ละตัว
และเมื่อมีการเพิ่มข้อมูลที่จำเป็นต้องขยายขนาด ก็จะสร้าง std::vector ใหม่ ต่อท้าย mData นี้

สำหรับตัวอย่างข้างต้น ขอยกตัวอย่างการเข้าถึงช่องต่าง ๆ ใน vector_no_move ดังนี้

ช่องของ vector_no_move ที่ต้องการเข้าถึง	**mData ที่เก็บค่าช่องนั้น**
0	mData[0][0]
3	mData[0][3]
4	mData[1][0]
6	mData[2][0]
9	mData[2][3]

เนื้อหาหลักสำหรับข้อ Vector Some Move

ต่อมาต้องการที่จะสร้าง CP::vector_some_move โดยภายในเหมือนกับ CP::vector_no_move ทุกประการเว้นแต่ว่ามีฟังก์ชั่น void insert(int index, vector<T> &value) เพิ่มขึ้นมา

โดยสิ่งที่ void insert(int index, vector<T> &value) ทำคือนำข้อมูลจากใน value ทั้งหมดไปแทรกที่ตำแหน่ง**ก่อนข้อมูลตำแหน่ง index** โดยเราสามารถแก้ไข mData ในรูปแบบใดก็ได้ตราบใดที่ให้ผลลัพธ์ที่ถูกต้องหลังเรียกคำสั่ง insert

ข้อบังคับ

- โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ Code::Blocks ให้ซึ่งในไฟล์โปรเจ็คดังกล่าวจะมีไฟล์ vector_some_move.h, main.cpp และ student.h อยู่ ให้นิสิตเขียน code เพิ่มเติมลงในไฟล์ student.h เท่านั้น และการส่งไฟล์เข้าสู่ระบบ grader ให้ส่งเฉพาะไฟล์ student.h เท่านั้น
- ในไฟล์ student.h ดังกล่าวจะต้องไม่ทำการอ่านเขียนข้อมูลใด ๆ ไปยังหน้าจอหรือคีย์บอร์ดหรือไฟล์ใด ๆ
- **สามารถ include library ใดเพิ่มก็ได้ในไฟล์ student.h แต่จะต้องไม่มีการใช้ compiler directive อื่น grader จะไม่ทำการตรวจสอบเรื่องนี้ แต่จะมีการตรวจสอบภายหลัง การผิดเงื่อนไขจะได้คะแนนในข้อนี้เป็น 0**

คำอธิบายฟังก์ชั่น main()

main จะสร้าง CP::vector_some_move<int> v โดยในตอนแรกมีขนาด 0 และไม่มีข้อมูล

จากนั้น main จะอ่านข้อมูลมาหลายบรรทัดตามรูปแบบนี้

- บรรทัดแรกประกอบด้วยจำนวนเต็ม n m และ pos โดย n คือจำนวนครั้งที่จะ resize vector, m คือจำนวนข้อมูลที่จะ insert และ pos คือตำแหน่งใน v ที่จะ insert
- บรรทัดที่สองประกอบด้วยจำนวนเต็ม n ตัว โดยแต่ละตัวเป็นขนาดที่จะเพิ่มให้ v (นั่นคือทำให้ v มีขนาดหลัง resize แต่ละครั้งเป็น v.size() + จำนวนเต็มตัวนั้น)
- บรรทัดที่สามประกอบด้วยจำนวนเต็ม v.size() ตัว โดย v.size() คือขนาดหลังจากการที่ v ถูก resize n ครั้ง โดยบรรทัดที่ 3 คือข้อมูลตั้งแต่ index 0 ถึง v.size() -1 ของ v
- บรรทัดที่สี่ประกอบด้วยจำนวนเต็ม m ตัว โดยคือแต่ละตัวคือข้อมูลที่จะ insert เข้าไปใน v ที่ตำแหน่ง pos ตามลำดับ

เมื่ออ่านข้อมูลเสร็จ main จะพิมพ์ค่า v ก่อนการ insert และ v หลังการ insert

ตัวอย่าง

ข้อมูลนำเข้า	ข้อมูลส่งออก
4 2 1 4 3 1 2 1 2 3 4 5 6 7 8 9 10 10 9	--- BEFORE INSERT --- 1 2 3 4 5 6 7 8 9 10 --- AFTER INSERT --- 1 10 9 2 3 4 5 6 7 8 9 10

รับประกันว่าการทดสอบใน grader

- ขนาดของ vector_some_move สุดท้ายจะไม่เกิน 3,500,000
- ข้อมูลทดสอบถูกออกแบบมาให้สามารถทำงานได้ใน 1 วินาที

ชุดข้อมูลทดสอบ

กำหนดให้ x เป็นจำนวนครั้งที่เรียกคำสั่ง insert และ sz หมายถึงขนาดของ vector ที่จะทำการ insert

- 10% x <= 200, sz <= 100, pos = v.size()
- 20% x <= 2,500, sz <= 100
- 30% x <= 30,000, sz <= 10 และรับประกันว่า pos = ผลรวมขนาดของ vector ที่แทรกก่อนหน้าทั้งหมด
- 40% x <= 30,000, sz <= 100

คำแนะนำ

- คลาส CP::vector_some_move มีฟังก์ชันชื่อ debug อยู่
  ซึ่งสามารถเรียกใช้เพื่อพิมพ์ค่าต่าง ๆ ใน vector_some_move ออกมาได้

ข้อควรระวัง

- การ insert มีผลต่อ operator[] ดังนั้นจึงจำเป็นต้องมีการแก้ไข vector<int> aux เพื่อให้ operator[] ทำงานได้ถูกต้อง โดย aux[i] จะหมายถึงผลรวมขนาดของ vector ตั้งแต่ mData[0] จนถึง mData[i]

***** main ที่ใช้จริงใน grader นั้นจะแตกต่างจาก main ที่ได้รับในไฟล์โปรเจ็คเริ่มต้น *****