Vector No Move

โดยปรกติแล้ว เมื่อ CP::vector เต็ม (กล่าวคือ mSize == mCap) แล้วจำเป็นต้องเพิ่มข้อมูลเข้าไปนั้น CP::vector จะทำการขยายขนาด mData โดยการจอง dynamic array ใหม่ ให้มีขนาดอย่างน้อย 2 เท่าของ mCap แล้วจึงย้ายข้อมูลทั้งหมดใน mData เก่าไปยัง dynamic array ตัวใหม่ที่พึ่งสร้างขึ้น ขั้นตอนนี้จะเสียเวลาในการย้ายข้อมูล

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

ตัวอย่างเช่น หาก mCap = mSize = 4 และ ใช้ array 1 ตัวในการเก็บข้อมูลเหล่านั้น และต้องการเพิ่มข้อมูลเข้าไป 2 ตัวแล้ว vector ใหม่นี้จะสร้าง array ใหม่เพียง 2 ช่อง ทำให้ vector ใหม่นี้ มี 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 แบบใหม่นี้ เพื่อความสะดวก vector ใหม่นี้จะมีฟังก์ชันเพียงแค่ constructor, push_back, resize และ operator[] เท่านั้นโดยยังไม่รองรับการใช้งาน iterator หรือฟังก์ชันอื่น ๆ ส่วนข้อมูลที่คลาสนี้เก็บได้แก่ mSize และ mCap นั้นการใช้งานเหมือน vector ทั่วไป แต่ mData จะถูกประกาศเป็น std::vector<std::vector<T>> mData แทน กล่าวคือ mData[i] จะเป็น vector<T> ซึ่งแทน array แต่ละตัว และเมื่อมีการเพิ่มข้อมูลที่จำเป็นต้องขยายขนาด ก็จะสร้าง std::vector ใหม่ต่อท้าย 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]

จงเขียนฟังก์ชัน operator[](int idx) ของคลาสนี้ เพื่อคืนค่าช่องหมายเลข idx จากข้อมูล mData ดังกล่าว

นอกจากนี้ รับประกันว่า การใช้งานคลาส CP::vector_no_move นี้จะเรียก resize ด้วยขนาดที่เพิ่มขึ้นเสมอ และเนื่องจากคลาสนี้ไม่มีฟังก์ชันในการลบข้อมูล เราจึงไม่จำเป็นต้องกังวลว่าขนาดของข้อมูลจะลดลง และ รับประกันว่า การเรียก operator[](int idx) นั้น ค่า idx จะอยู่ในช่วง 0 ถึง mSize-1 เสมอ

ข้อบังคับ

- โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ Code::Blocks ให้ ซึ่งในไฟล์โปรเจ็คดังกล่าวจะมีไฟล์ vector.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

คำแนะนำ

- ฟังก์ชัน operator[] นั้นควรจะทำงานได้อย่างรวดเร็ว เพื่อทำสิ่งนี้ให้ได้ เราอาจจะต้องมีการทำงานบางอย่างตอนที่เราขยายขนาดข้อมูล ในคลาสนี้ มีฟังก์ชัน expand_hook() ที่จะถูกเรียกทุกครั้งที่เกิดการ expand และมี data member เป็น vector<int> aux ที่เราสามารถใช้งานได้ การที่จะทำงานให้ถูกต้องสำหรับคลาสนี้นั้นไม่จำเป็นต้องใช้งาน expand_hook หรือ aux แต่หากจะทำงานให้ได้รวดเร็ว อาจจะจำเป็นต้องใช้ข้อมูลดังกล่าว
- คลาสนี้มีฟังก์ชันชื่อ debug อยู่ ซึ่งสามารถเรียกใช้เพื่อพิมพ์ค่าต่าง ๆ ใน vector_no_move ออกมาได้

คำอธิบายฟังก์ชัน main()

main จะอ่านข้อมูลมาหลายบรรทัด ตามรูปแบบนี้

- บรรทัดแรกประกอบด้วยจำนวนเต็ม n และ m
  - โปรแกรมจะสร้าง CP::vector_no_move<int> มา 1 ตัวคือ v
- หลังจากนั้นอีก n บรรทัดจะเป็นข้อมูลการกระทำต่าง ๆ ต่อคลาส vector_no_move โดยที่แต่ละบรรทัดประกอบด้วยตัวเลขจำนวนเต็มสองตัวคือ t และ a
  - หาก t มีค่าเป็น 1 โปรแกรมจะทำการ push_back ใส่ v ด้วย a
  - หาก t มีค่าเป็น 2 โปรแกรมจะทำการ resize v ด้วย a (รับประกันว่าค่า a จะมากกว่า v.size() เสมอ)
  - หาก t มีค่าเป็น 3 โปรแกรมจะทำการเปลี่ยนค่าช่อง a ใน v ให้เป็น 0

เมื่อกระทำการตามคำสั่งต่าง ๆ เสร็จแล้ว โปรแกรมจะทำการพิมพ์ค่าใน vector ทั้งหมดทุกช่องออกมาซ้ำกัน m รอบ

ชุดข้อมูลทดสอบ

- 20% n, m <= 10 และมีแต่คำสั่งประเภท t = 1
- 20% n, m <= 100 และมีแต่คำสั่งประเภท t = 1 และ 2
- 20% n <= 1,000 และ m <= 100
- 40% n <= 10,000 และ m <= 1,000

ข้อควรระวัง

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