

## Erase Many from Vector

คลาส `CP::vector` นั้นมีฟังก์ชันสำหรับลบข้อมูลเพียงฟังก์ชันเดียว คือ `erase(iterator it)` สำหรับโจทย์ข้อนี้จึงเขียนฟังก์ชันเพิ่มเติมความสามารถในการลบข้อมูลของ `CP::vector` โดยให้เพิ่มฟังก์ชัน `erase_many( vector<size_t> pos)` ซึ่งจะต้องลบข้อมูลทุกตัวที่อยู่ในตำแหน่ง `pos` ให้หมด ตัวอย่างเช่น ถ้า `v = <A, B, C, D, E, F, G, H>` และให้ `pos = <0,3,4>` เมื่อสั่ง `v.erase_many(pos)` แล้ว จะทำให้ `v` เหลือข้อมูลเป็น `<B, C, F, G, H>` (ซึ่งหมายความว่าข้อมูลตัวที่ 0, 3, 4 หายไป) ฟังก์ชันนี้จะต้องคืนจำนวนข้อมูลที่ลบไปได้

รับประกันว่าค่าตำแหน่ง `pos` นั้นจะเรียงลำดับจากน้อยไปมาก และค่าดังกล่าวจะมีค่าอยู่ในช่วง 0 ถึง `v.size()-1` แน่นอน

ให้ระวังว่าการลบข้อมูลของ `vector` นั้นใช้เวลาขึ้นอยู่กับตำแหน่งที่ทำการลบ นิสิตจะต้องหาทางลบข้อมูลให้ใช้เวลาไม่นานเกินไป สำหรับโจทย์ข้อนี้ test data 2 อันสุดท้ายมีขนาดใหญ่มาก แนะนำให้เลือกวิธีที่เขียนโปรแกรมไม่ยากมาก ที่ทำให้ได้อย่างน้อย 50-80 คะแนนก่อน

## ข้อบังคับ

โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ `code::block` ให้ ซึ่งในโปรเจ็คดังกล่าวจะมีไฟล์ `vector.h`, `main.cpp` และ `student.h` อยู่ ให้นิสิตเขียน code เพิ่มเติมลงไปไฟล์ `student.h` เท่านั้น และการส่งไฟล์ขึ้น grader ให้ส่งเฉพาะไฟล์ `student.h`

## คำอธิบายฟังก์ชัน main()

`main` จะทำการเรียกใช้ฟังก์ชันต่าง ๆ ของ `vector` โดยโปรแกรมจะเริ่มจาก `vector` ว่าง 1 อัน และเรียกใช้งาน `vector` ดังกล่าวตามข้อมูลคำสั่งที่ได้รับจาก keyboard มาทีละบรรทัด แต่ละบรรทัดนั้นจะมีการทำงานต่าง ๆ ซึ่งขึ้นอยู่กับตัวอักษรตัวแรกในบรรทัด โดยที่ `a` เป็นการเพิ่มข้อมูล เข้าไปใน `vector`, `e` เป็นการเรียกใช้บริการ `erase_many` ส่วน `p` จะเป็นการพิมพ์ข้อมูลใน `vector` ออกมา และ `q` เป็นการจบการทำงาน

บรรทัดที่มีคำสั่ง `a` และ `e` มีรูปแบบดังนี้

- คำสั่ง `a` จะตามด้วยตัวเลข 1 ตัว สมมติให้มีค่าเป็น `n` และตามด้วยตัวเลขอีก `n` ตัว ซึ่งตัวเลข `n` ตัวนั้นจะถูกนำไปใส่ใน `vector` ตามลำดับ
- คำสั่ง `e` จะตามด้วยตัวเลข 1 ตัว สมมติให้มีค่าเป็น `n` และตามด้วยตัวเลขอีก `n` ตัว ซึ่งตัวเลข `n` ตัวนั้นคือค่าของ `pos` ที่จะนำไปใช้เป็น parameter ของบริการ `erase_many`

### ตัวอย่าง

ข้อมูลที่พิมพ์เข้าทาง keyboard	ข้อมูลที่เป็นผลจากการทำงานของโปรแกรม
a 10 0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9
p	0 2 3 4 5 6 7 8 9
e 1 1	2 6 8 9
p	2 6 8 9 100 200 300
e 5 0 2 3 4 6	
p	
a 3 100 200 300	
p	
q	