K-Ary Heap

คลาส CP::priority_queue นั้นใช้โครงสร้างข้อมูลที่ชื่อว่า binary heap ซึ่งเป็น full binary tree แบบหนึ่ง กล่าวคือ ปมหนึ่งปมในต้นไม้นั้นจะมีลูกไม่เกิน 2 ลูก และทุก ๆ ชั้นความสูงของต้นไม้มีปมเต็มทุกชั้น ยกเว้นเฉพาะชั้นที่ลึกที่สุดที่ปมจะอาจจะไม่เต็มได้ แต่ปมทุกปมต้องอยู่เต็มจากซ้ายไปขวา นอกจากนี้ binary heap นั้นบังคับให้ค่าในปมใด ๆ นั้นต้อง "มากกว่า" ค่าในปมลูกเสมอ จากข้อกำหนดดังกล่าวทำให้เวลาที่ใช้ในบริการ push และ pop ของ binary heap นั้นมีจำนวนครั้งที่ใช้เปรียบเทียบค่าของสมาชิกเป็น $O(\log_2 N)$ เมื่อ N คือจำนวนข้อมูลใน binary heap

เราสามารถทำให้ binary heap ใช้จำนวนครั้งในการเปรียบเทียบค่าของสมาชิกใน heap น้อยลงได้โดยการเปลี่ยนไปใช้ K-ary heap แทน (กล่าวคือใช้ต้นไม้ที่มีปมไม่เกิน K ปม) โดยที่กฎต่าง ๆ ของ binary heap นั้นยังคงใช้กับ k-ary heap เหมือนเดิม ซึ่งจะทำให้จำนวนครั้งในการเปรียบเทียบข้อมูลกลายเป็น $O(\log_K N)$

จากคลาส CP::priority_queue นั้น จงเขียนฟังก์ชัน fixup(size_t idx) และ fixdown(size_t idx) ซึ่งเป็นฟังก์ชันสำหรับปรับตำแหน่งของสมาชิกใน heap โดยที่ fixup จะปรับตำแหน่งของสมาชิก ณ ตำแหน่ง idx ให้วิ่งขึ้นเข้าไปในทิศใกล้รากมากขึ้นจนอยู่ในตำแหน่งที่ถูกต้อง โดยสลับปมพ่อลงมาแทน ส่วน fixdown นั้นจะปรับตำแหน่งของสมาชิก ณ ตำแหน่ง idx ให้วิ่งลงไปไกลรากมากข้นจนอยู่ในตำแหน่งที่ถูกต้อง โดยสลับเอาสมาชิกตัวที่ "มากที่สุด" ในบรรดาลูก ๆ ทั้ง K ตัวขึ้นมาแทน

ข้อบังคับ

ในข้อนี้ กำหนดให้ K มีค่าเป็น 4 โจทย์ข้อนี้จะมีไฟล์โปรเจ็คของ code::block ให้ ซึ่งในโปรเจ็คดังกล่าวจะมีไฟล์ priority_queue.h, main.cpp และ student.h อยู่ ให้นิสิตเขียน code เพิ่มเติมลงไปในไฟล์ student.h เท่านั้น และการส่งไฟล์ขึ้น grader ให้ส่งเฉพาะไฟล์ student.h

รับประกันว่าข้อมูลที่ใส่ไปใน Heap นั้นจะไม่เคยซ้ำกันเลย ดังนั้น นิสิตไม่ต้องกังวลเรื่องข้อมูลซ้ำกัน

ข้อควรระวัง!!!

- การเปรียบเทียบข้อมูลใน heap นั้นต้องใช้ mLess ในการเปรียบเทียบ อย่าใช้ < ในการเปรียบเทียบ
- สามารถดูตัวอย่าง ฟังก์ชัน fixup, fixdown ของ binary heap ได้จากข้อ heap_order2
- main.cpp และ priority_queue.h ใน grader จะไม่เหมือนกับ main.cpp และ priority_queue.h ที่แจกให้

คำอธิบายฟังก์ชัน main()

main จะทำการอ่านข้อมูลจาก keyboard ซึ่งระบุถึงการทำงานกับ Heap แล้วกระทำการกับ heap ตามคำสั่งที่ได้รับมา ในแต่ละบรรทัดจะมีคำสั่งหนึ่งคำสั่ง โดยมีคำสั่งทั้งหมด 4 คำสั่งคือ i สำหรับการใส่ข้อมูล x สำหรับการลบค่าข้อมูล “มากสุด” และ q สำหรับการหยุดการทำงาน ตัวอย่างเช่น

```
i 5
i 3
i 7
i 1
i 2
i 13
x
q
```

จะทำการ ใส่ข้อมูล 5, 3, 7, 1, 2 เข้าไปใน heap ตามลำดับ และ จากนั้นก็ทำการลบค่าที่มากสุด (13) แล้วจบการทำงาน แล้ว main จะทำการพิมพ์ค่าออกมาสองบรรทัด โดยบรรทัดแรกคือ ค่าทั้งหมดใน mData ของ pq ส่วนบรรทัดที่สองจะพิมพ์ผลลัพธ์จากการ pop ค่าออกมาจาก pq ทีละค่า

ตัวอย่าง

ข้อมูลนำเข้า	**ข้อมูลส่งออก**
i 5 i 3 i 2 i 1 q	5 3 2 1 5 3 2 1

ข้อมูลนำเข้า	**ข้อมูลส่งออก**
i 1 i 2 i 3 i 4 i 5 x x q	3 1 2 3 2 1