Lab01: Arrays

22 มกราคม 2568

อย่าเหลี่ยม (No Brackets)

แลบนี้เราจะมาฝึกความคุ้นชินในเรื่องของการเข้าถึงอาร์เรย์โดยใช้ตัวชี้ (Pointer) เพราะเรื่องนี้จะถูกใช้ในเรื่อง Struct, Linked list, stack, queue (เป็นเรื่องที่จะได้เรียนในสัปดาห์ถัด ๆ ไป) เราจะมาฝึกการใช้ Pointer ให้คุ้นมือกันก่อน

โจทย์ข้อนี้จะท้าทายให้นักศึกษาเขียนโปรแกรมที่จะพิมพ์อาร์เรย์ออกมา โดยเราจะไม่ให้ใช้**วงเล็บเหลี่ยม (ห้ามมีวงเล็บเหลี่ยมใน Source Code ใด ๆ ทั้งสิ้น -> วงเล็บเหลี่ยมก็คือ Brackets [])** ดังนั้นนักศึกษาต้องเขียนอาร์เรย์ในรูปแบบที่ต่างออกไป

เมื่อเรารับค่าเอาร์เรย์เข้ามาแล้ว เราจะพิมพ์อาร์เรย์ตามโหมดที่เราตั้ง ซึ่งก็คือ

- `0`: เราจะพิมพ์สมาชิกในตำแหน่ง**คู่**ของอาร์เรย์
- `1`: เราจะพิมพ์สมาชิกในตำแหน่ง**คี่**ของอาร์เรย์

โดยให้อาร์เรย์เริ่มที่ตำแหน่ง `'0'` และหากว่าไม่มีสมาชิกที่เข้าเงื่อนไขที่สามารถพิมพ์ออกมาเลย ให้พิมพ์ว่า **none**

ข้อมูลนำเข้า (Input)

บรรทัดที่ 1	จำนวนเต็ม n โดยที่ $1 \leq n \leq 10,000$ บอกจำนวนสมาชิกในอาร์เรย์
บรรทัดที่ 2	จำนวนเต็มทั้งหมด n จำนวน โดยคั่นแต่ละตัวด้วยช่องว่างหนึ่งช่อง
บรรทัดที่ 3	จำนวนเต็มที่บอกโหมด มีค่าเป็นได้แค่ 0 หรือ 1 เท่านั้น

ข้อมูลส่งออก (Output)

บรรทัดที่ 1	ลำดับของอาร์เรย์ตามโหมดที่พิมพ์ แต่ละตัวคั่นด้วยช่องว่างหนึ่งช่อง

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
5 1 1 1 2 3 0	1 1 3
10 6 5 8 4 9 7 8 5 1 3 1	5 4 7 5 3
1 2 1	none

จัดการอาร์เรย์ (Array Manipulation)

หลังจากนี้ไปนักศึกษาจะได้ใช้การประกาศอาร์เรย์แบบ Dynamic Allocation เป็นส่วนใหญ่ ซึ่งมีข้อดีตรงที่นักศึกษายังไม่จำเป็นต้องรู้จำนวนพื้นที่ว่างที่แน่ชัดในการทำงาน

งานนี้จะให้นักศึกษาเขียนโปรแกรมเพื่อทำ Operation ต่าง ๆ บนอาร์เรย์ดังนี้

1. แทรก (Insert) ค่าที่ตำแหน่งใด ๆ ในอาร์เรย์
2. ลบ (Delete) ค่าที่ตำแหน่งใด ๆ ในอาร์เรย์
3. รวม (Merge) สองอาร์เรย์เข้าเป็นอาร์เรย์อันเดียว
4. หากว่าเราจะเข้าถึงตำแหน่งที่ไม่ได้จัดไว้ให้ (Un-allocated memories) ให้พิมพ์ว่า `Index out of bounds`

โดยมีฟังก์ชันให้นักศึกษา Implement ดังนี้

1. `insertArray(int* arr, int size, int index, int value)`: เป็นการแทรก (Insert) ค่าเข้าไปที่ตำแหน่งนั้น ๆ ในอาร์เรย์ และให้ Return อาร์เรย์ตัวใหม่ออกมา
2. `deleteArray(int* arr, int size, int index)`: เป็นการลบ (Delete) ค่าออกจากอาร์เรย์และให้ Return อาร์เรย์ตัวใหม่ออกมา
3. `mergeArray(int* arr1, int size1, int* arr2, int size2)`: เป็นการรวม (Merge) อาร์เรย์สองตัวเข้าด้วยกันและ Return อาร์เรย์ตัวใหม่ออกมา
4. `printArray(int *arr, int size)`: เป็นการพิมพ์ค่าทั้งหมดที่อยู่ในอาร์เรย์นั้น ๆ ออกมา

อย่าลืมใช้ `malloc()` ในการจองหน่วยความจำ นอกจากนี้เรายังสามารถใช้ `realloc()` ในการเปลี่ยนขนาดการจองหน่วยความจำ และเมื่อจบโปรแกรม อย่าลืม De-allocate หน่วยความจำด้วย

ข้อมูลนำเข้า (Input)

บรรทัดที่ 1	จำนวนเต็มบวก n_1 โดยที่ $1 \leq n_1 \leq 10,000$ แทนจำนวนสมาชิกในอาร์เรย์ตัวที่ 1
บรรทัดที่ 2	จำนวนเต็ม n_1 จำนวน แทนสมาชิกแต่ละตัวในอาร์เรย์ตัวที่ 1 คั่นแต่ละตัวด้วยช่องว่างหนึ่งช่อง
บรรทัดที่ 3	จำนวนเต็มบวก n_2 โดยที่ $1 \leq n_2 \leq 10,000$ แทนจำนวนสมาชิกในอาร์เรย์ตัวที่ 2
บรรทัดที่ 4	จำนวนเต็ม n_2 จำนวน แทนสมาชิกแต่ละตัวในอาร์เรย์ตัวที่ 2 คั่นแต่ละตัวด้วยช่องว่างหนึ่งช่อง
บรรทัดที่ 5	จำนวนเต็ม i โดยที่ $i \leq n_1$ แทนตำแหน่งที่ต้องการแทรก (Insert) ในอาร์เรย์ตัวแรก
บรรทัดที่ 6	ค่าที่ต้องการแทรกในตำแหน่งที่ i ของอาร์เรย์ตัวแรก
บรรทัดที่ 7	จำนวนเต็มบวก j โดยที่ $j \leq n_1$ แทนตำแหน่งที่ต้องการลบ (Delete) ในอาร์เรย์ตัวแรก

ข้อมูลส่งออก (Output)

บรรทัดที่ 1	อาร์เรย์ที่ได้จากการแทรก (Insert)
บรรทัดที่ 2	อาร์เรย์ที่ได้หลังจากการลบ (Delete)
บรรทัดที่ 3	อาร์เรย์ที่ได้หลังจากการรวม (Merge)

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
5 1 2 3 4 5 5 6 7 8 9 10 0 0 0	0 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 6 7 8 9 10
2 1 2 2 3 4 0 5 3	5 1 2 Index out of bounds 5 1 2 3 4
3 1 2 3 3 4 5 6 1 50 2	1 50 2 3 1 50 3 1 50 3 4 5 6