

FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 INTRODUCTIONS TO DATA STRUCTURE
Year II, First Semester, Midterm Examination, Oct 6, 2022 08:30-11:30

ชื่อ-นามสกุล.....เลขประจำตัว.....ตอนเรียนที่.....เลขที่ใน CR58.....

หมายเหตุ

1. ข้อสอบมีทั้งหมด 11 ข้อ ในกระดาษคำถามคำตอบ 10 หน้า
2. ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ
3. ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ
4. ห้ามการหยิบยื่นสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยื่นให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. **นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับสัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่า
นิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และ
อาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

* ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ *

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....

วันที่.....

- ใช้ดินสอเขียนคำตอบได้
- ให้เขียนเลขที่ในใบเซ็นชื่อเข้าสอบทุกหน้า
- หากพื้นที่สำหรับเขียนคำตอบไม่เพียงพอ ให้เขียนไว้ด้านหลังของหน้านั้น ห้ามเขียนข้ามไปหน้าอื่น และให้ระบุไว้ในพื้นที่สำหรับเขียนคำตอบว่า “มีต่อด้านหลัง”

--	--	--	--	--	--	--	--	--	--

--	--	--

1. (8 คะแนน) ตอบคำถามต่อไปนี้สั้น ๆ ว่า แต่ละปัญหาต้องมีที่เก็บข้อมูลประเภทใด

1.0 ต้องการเก็บรายชื่อนิสิตคณะวิศวกรรมศาสตร์ทุก ๆ รุ่น แต่ละรุ่นมีหมายเลขรุ่นกำกับ เพื่อเขียนเมทอด

`int getClassID(string name)` ที่คืนหมายเลขรุ่นของคนชื่อ `name`

ตอบ: `map<string,int>` *key* คือชื่อ *mapped value* คือหมายเลขรุ่น (ข้อนี้เป็นตัวอย่าง)

1.1. ต้องการเก็บข้อมูล เพลงต่าง ๆ ในโทรศัพท์มือถือ ซึ่งเพลงหนึ่งเพลงจะประกอบด้วย ชื่อเพลง ชื่ออัลบั้ม และชื่อนักร้อง โดยจะต้องสามารถค้นหาได้เร็วโดยใช้ชื่อเพลง

1.2. ต้องการเก็บข้อมูล สำหรับการ์ด (การ์ดหนึ่งใบแทนด้วยสตริงที่บอกชื่อของการ์ด) ที่ใช้ในระหว่างเล่นการ์ดเกม โดยสำหรับ จะต้องเกิดจากการ์ดวางทับกัน ผู้เล่นจะต้องสามารถจั่วการ์ดจากด้านบนสุดของสำหรับ และสามารถเอาการ์ดใบบนสุดไปไว้ล่างสุดของกองการ์ดได้

1.3. ต้องการเก็บข้อมูลสถิติการส่งการบ้านของนักเรียนหลายคน โดยข้อมูลนี้ต้องสามารถถูกนำไปค้นได้โดยใช้ ชื่อวิชา (string), และเลขที่แบบฝึกหัด (int) โดยจะได้ผลลัพธ์เป็น ข้อมูล ชื่อนักเรียน (string) และคะแนนนักเรียน (int) ของนักเรียนมากกว่าหนึ่งคน ให้แบบฝึกหัดหนึ่งของวิชาหนึ่ง ๆ นั้นนักเรียนแต่ละคนส่งได้ครั้งเดียว

1.4. ต้องการเก็บข้อมูล ผลคะแนนตารางแข่งขันฟุตบอล โดยแต่ละแถวของตารางประกอบด้วย ชื่อทีม (string), คะแนน (int), ผลต่างประตู (int), จำนวนประตูที่ยิงได้ (int) โดยแถวในตาราง จะต้องถูกทำให้เรียงแถวตาม คะแนน, ผลต่าง, จำนวนประตูที่ยิงได้ , และชื่อทีม ตามลำดับ ในการอัปเดตค่าแต่ละครั้ง

2. (5 คะแนน) จงเขียนส่วนของโปรแกรมที่ทำการพิมพ์เฉพาะข้อมูลประเภท string ทั้งหมดที่เก็บอยู่ในตัวแปร `x` ซึ่งถูกประกาศเป็นประเภทดังต่อไปนี้ (ให้ถือว่า `x` สามารถแก้ไขได้ ไม่จำเป็นต้องทำให้ `x` มีค่าคงเดิมหลังพิมพ์)

2.1. `vector<pair<int, string>> x;`

2.2. `stack<vector<string>> x;`

2.3. `map<int, pair<int, pair<string, int>>> x;`

2.4. `vector<map<vector<string>, stack<int>>> x;`

2.5. `set<priority_queue<pair<bool, string>>> x;`

3. (6 คะแนน) จงเติมโค้ดลงในช่องว่าง เพื่อให้ฟังก์ชันที่มีทำงานตรงตามที่โจทย์กำหนด

- 3.1. มีคนเข้าคิวซื้อตั๋วสวนสนุก โดยผู้เข้าคิวแต่ละคนจะมีคู่ข้อมูลที่ระบุ (ชื่อ, Boolean flag ว่าจองทางเว็บแล้วหรือไม่) เราต้องการเขียนฟังก์ชันที่จะแยกคนที่จองทางเว็บ (มีข้อมูลที่สองในคู่ข้อมูล เป็น true) ออกจากคิวแล้วเอามาทำเป็นอีกคิวหนึ่ง โดยให้คืนค่าคิวที่มีแต่คนที่จองทางเว็บเท่านั้น ส่วนคิวเดิม จะต้องลบคนที่จองทางเว็บออกให้หมด จงเติมโค้ดที่จะทำให้ฟังก์ชันนี้ทำงานถูกต้อง

```
queue<pair<string,bool>> split(queue<pair<string,bool>> &q) { // รับคิวที่มีคนบนกันหมด

    _____;

    queue<pair<string,bool>> q2;
    for(int i =0; i < _____ ; i++) {
        pair<string,bool> x = q.front();
        if(_____) { // กรณีที่เป็นคนที่จองทางเว็บ
            _____;
            q.pop();
        } else {
            q.pop();
            _____;
        }
    }
    _____;
}
```

- 3.2. เราต้องการจำลอง การเข้าคิวซื้อตั๋วหนัง ของโรงภาพยนตร์หนึ่ง ซึ่งมีช่องขายตั๋วอยู่ n ช่อง ช่องขายตั๋วนั้น จำลองด้วย vector<int> v ซึ่ง v[i] ระบุเวลาที่คนขายตั๋วช่องนั้น ๆ บริการลูกค้า (มีหน่วยเป็นนาที และ v[i] > 0) และมีคิวของลูกค้า จำลองด้วย queue<int> q ซึ่งค่าใน queue ระบุหมายเลขของลูกค้า ให้เวลาเริ่มต้นของการซื้อตั๋ว เมื่อมีคน n คน เริ่มที่นาทีที่ 0 ตัวอย่าง ให้ v = {2,1,3,2,1} และคิวมีลูกค้าหมายเลข 1 ถึง 12 ตามลำดับ การขายตั๋วจะดำเนินดังนี้

- ในนาทีที่ 0 ลูกค้า n คนแรกจากคิว จะเข้ารับบริการแต่ละช่อง (ลูกค้าหมายเลข 1-5 ได้รับบริการ)
 - ในนาทีที่ 1 ช่อง 1 และช่อง 4 จะว่างให้บริการอีก (ลูกค้าหมายเลข 6,7 ได้รับบริการ)
 - ในนาทีที่ 2 ช่อง 0,1,3,4 จะว่างให้บริการ (ลูกค้าหมายเลข 8-11 ได้รับบริการ)
 - ในนาทีที่ 3 ช่อง 1,2,4 จะว่างให้บริการ (ลูกค้าหมายเลข 12 ได้รับบริการ)
- จงเขียนฟังก์ชันที่บอกเวลาที่ลูกค้าคนสุดท้ายในคิวได้รับบริการ (ในกรณีตัวอย่างข้างต้น ฟังก์ชันจะต้องคืนค่า 3)

```
int calculate_time(queue<int> q, vector<int> v) {
    int time = 0;
    while(_____) {
        for(int i =0; _____; i++) {
            if(_____) return time;
            if(_____) {
                q.pop();
            }
        }
        if(q.empty()) _____;
        time++;
    }
    _____;
}
```

4. (5 คะแนน) จงระบุ complexity ที่ถูกต้องที่สุดสำหรับส่วนของโปรแกรมดังต่อไปนี้เมื่อเทียบกับตัวแปร n

4.1	<pre>int f1(vector<int> &a, int e, int x) { int n = a.size(); int k = 0; for (int i = 0; i < n; i++) if (a[i] == e) k++; return k; }</pre>	
4.2	<pre>int f2(vector<int> &a) { int n = a.size(); for (int i = 0; i < n - 1; i++) for (int j = i; j < n; j++) if (a[i] == a[j]) return a[i]; }</pre>	
4.3	<pre>int f3(int n, int x) { // รับประกันว่า 0 <= x < n int sum = 0; for (int i = 0; i < n; i++) { sum = sum + i; int f = 0; while(f < n) { sum += f++; } if (i == x) return sum; } }</pre>	
4.4	<pre>void f4(int n) { // รับประกันว่า 0 < n int k = 10; priority_queue<int> pq; for (int i = 0; i < k; i++) pq.push(i); for (int i = 0; i < n; i++) { pq.push(k+1); pq.pop(); } }</pre>	
4.5	<pre>void f5(int n) { // รับประกันว่า 0 < n vector<int> v1; vector<vector<int>> v2; for (int i = 0; i < n; i++) { v1.insert(v1.begin(),0); v2.insert(v2.begin(),v1); } }</pre>	

5. (4 คะแนน) จงพิจารณาส่วนของโปรแกรมต่อไปนี้ ซึ่งต้องการ “ย้อนกลับลำดับ” (reverse) ข้อมูลใน CP::vector ตั้งแต่ช่องที่ระบุด้วย iterator it1 จนถึงช่องก่อนที่ระบุด้วย iterator it2 (รับประกันว่า it1 < it2 <= end()) ในส่วนของโปรแกรมนี้อาจมีที่ผิดพลาดอยู่ จงระบุบรรทัดที่ผิดพลาด พร้อมทั้งอธิบายว่าความผิดพลาดดังกล่าวคืออะไร

```
1: template <typename T>
2: class vector { //คลาส vector นี้ทำงานได้ตามปรกติทุกอย่าง มีเพียงฟังก์ชัน reverse เพิ่มเข้ามาเท่านั้น
3: void reverse(iterator it1, iterator it2) {
4:     int n = it2 - it1;
5:     int p = it1 - begin();
6:     for (int i = 0; i < n; i++) {
7:         insert(it2,*it1);
8:         erase(it1);
9:         it1 = mData + p;
10:        it2--;
11:    }
12: }
13:};
```

6. (4 คะแนน) หากเราเปลี่ยนฟังก์ชัน ensureCapacity ของคลาส CP::queue ให้เป็นดังด้านขวานี้ (กล่าวคือ เมื่อ queue เต็ม เราจะขยายขนาดที่ละ 1 ช่องแทนการขยายขนาดที่ละ 2 เท่าของ mCap) จงพิจารณาส่วนของโปรแกรมต่อไปนี้ ซึ่งเป็นการใช้งาน CP::queue<int> จงระบุค่าของ mCap, mFront, mSize ของ q เมื่อ a b และ c มีค่าตามที่กำหนด

```
void ensureCapacity(size_t capacity) {
    if (capacity > mCap) expand(capacity);
}
```

```
CP::queue<int> q;
for (int i = 0; i < a; i++) q.push(1);
for (int i = 0; i < b; i++) q.pop();
for (int i = 0; i < c; i++) q.push(2);
```

ข้อย่อย	mSize	mCap	mFront	a	b	c
(1)				1	0	9
(2)				18	13	6
(3)				20	10	40
(4)				8	4	9

7. (10 คะแนน) ในข้อย่อยต่อไปนี้ให้ตอบโดยเลือกคำตอบที่ถูกต้องที่สุด แต่ละข้อย่อยมีคะแนน 1 คะแนน หากไม่ตอบในข้อย่อยใด จะได้คะแนน 0 แต่ถ้าหากตอบผิดในข้อย่อยใด จะได้คะแนน -0.5 ต่อข้อ (คะแนนที่น้อยที่สุดที่เป็นไปได้ของข้อนี้คือ -5) ให้ตอบโดยการเขียนตัวเลือกที่ต้องการลงในตารางข้างล่างนี้

ข้อ 7.1	ข้อ 7.2	ข้อ 7.3	ข้อ 7.4	ข้อ 7.5	ข้อ 7.6	ข้อ 7.7	ข้อ 7.8	ข้อ 7.9	ข้อ 7.10

**** การทำเครื่องหมายบนตัวเลือกข้างล่างจะไม่นับเป็นการตอบ จะยึดการตอบจากการเติมตัวเลือกลงในตารางข้างบนนี้เท่านั้น ****

7.1. ข้อความใดต่อไปนี้ผิด

- ก. การลบข้อมูลตัวแรกออกจาก `std::vector<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$
- ข. การแทรกข้อมูลหนึ่งตัวเข้าไปที่ตำแหน่งแรกของ `std::vector<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$
- ค. การหาข้อมูลทีมากที่สุดใน `std::vector<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$
- ง. การลดขนาด `std::vector<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$

7.2. ข้อความใดต่อไปนี้ผิด

- ก. การลบข้อมูลหนึ่งตัวจาก `std::set<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(\log n)$
- ข. การเพิ่มข้อมูลหนึ่งตัวให้แก่ `std::set<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(\log n)$
- ค. การแสดงทุกข้อมูลใน `std::set<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(\log n)$
- ง. การหาข้อมูลทีมากที่สุดใน `std::set<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(\log n)$

7.3. ข้อความใดต่อไปนี้ผิด

- ก. การแสดงค่าข้อมูลทุกตัวใน `std::set<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$
- ข. การแสดงค่าข้อมูลทุกตัวใน `std::priority_queue<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$
- ค. การแสดงค่าข้อมูลทุกตัวใน `std::vector<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$
- ง. การแสดงค่าข้อมูลทุกตัวใน `std::queue<int>` ที่มีสมาชิก n ตัวใช้เวลา $O(n)$

7.4. ข้อใดต่อไปนี้ เป็น copy constructor ที่ทำงานได้ถูกต้องของ `CP::vector` ที่มีหัวฟังก์ชันเป็น

`vector(const vector<T>& a) { /* โค้ด */ }`

ก.	<code>mData = new T[mCap]; for (size_t i = 0; i < mSize; i++) { mData[i] = a[i]; } mSize = a.size(); mCap = a.mCap;</code>	ข.	<code>swap(mData, a.mData); swap(mCap, a.mCap); swap(mSize, a.mSize);</code>
ค.	<code>mData = new T[a.mCap]; for (size_t i = 0; i < a.size(); i++) mData[i] = a.mData[i]; mSize=a.mSize; mCap=a.mCap;</code>	ง.	<code>this = &a;</code>

7.5. ข้อใดต่อไปนี้ เป็น copy constructor ที่ทำงานได้ถูกต้องของ `CP::queue` ที่มีหัวฟังก์ชันเป็น

`queue(const queue<T>& a) { /* โค้ด */ }`

ก.	<code>mData = new T[a.mCap]; mCap = a.mCap; mSize = a.mSize; mFront = 0; for (size_t i = 0; i < a.mCap; i++) { mData[i] = a.mData[i];</code>	ข.	<code>mData = new T[a.mCap]; mCap = a.mCap; mSize = a.mSize; mFront = a.mFront; for (size_t i = 0; i < a.mSize; i++) { mData[i] = a.mData[i];</code>
----	---	----	---

	}		}
ค.	<pre>mData = new T[a.mCap]; mCap = a.mCap; mSize = a.mSize; mFront = a.mFront; for (size_t i = 0; i < a.mCap; i++) { mData[i] = a.mData[(i+a.mFront) % a.mCap]; }</pre>	ง.	<pre>mData = new T[a.mCap]; mCap = a.mCap; mSize = a.mSize; mFront = a.mFront; for (size_t i = 0; i < a.mSize; i++) { mData[(i+mFront) % mCap] = a.mData[(i+a.mFront) % a.mCap]; }</pre>

- 7.6. ให้ `v` เป็น `vector<int>` ที่มีขนาด 1,000,000 ตัว โดยที่ `v[i]` มีค่าเป็น `i` ข้อใดต่อไปนี้จะใช้เวลามากที่สุด
- `v.lower_bound(124);`
 - `find(v.begin(), v.end(), 1);`
 - `int x = *(v.begin() + v.size() / 2);`
 - `v.erase(v.end() - 1);`
- 7.7. สำหรับคลาส `CP::queue` นั้น ข้อใดต่อไปนี้เป็นจริงเสมอ
- `mCap - mSize < mFront`
 - `mSize % mCap == mFront`
 - `mCap - mFront == mSize % mCap`
 - `mData + mCap > &mData[mFront]`
- 7.8. ให้ `v` เป็น `CP::queue<int>` ที่ถูกสร้างขึ้นมาแล้ว ข้อใดต่อไปนี้อาจเป็นการเรียกใช้งานช่องข้อมูลที่ไม่ได้จองไว้
- `*(mData + 0)`
 - `*(mData + mFront)`
 - `mData[mCap - 1]`
 - `mData[mCap - mSize]`
- 7.9. ให้ `find_in_pq(std::priority_queue<int> &pq)` เป็นฟังก์ชันที่ตรวจสอบว่าใน `pq` นั้นมีค่า `x` อยู่ภายในหรือไม่ (โดยไม่จำเป็นต้องทำให้ `pq` มีค่าคงเดิมหลังการหา) และ `find_in_pq` นั้นถูกเขียนให้ทำงานเร็วที่สุดเท่าที่ทำได้โดยการใช้งาน `priority_queue` แบบปกติ ข้อใดต่อไปนี้เป็นระดับเวลาการทำงานของ `find_in_pq` ที่ถูกต้องที่สุด
- $O(n)$
 - $O(n \log n)$
 - $\Theta(n)$
 - $\Theta(n \log n)$
- 7.10. ให้ดูส่วนของโปรแกรม radix sort ด้านขวามือนี้ หากเราเปลี่ยน `queue` เป็น `stack` (พร้อมเปลี่ยน `front()` เป็น `top()`) แล้วให้ `data` มีข้อมูลอย่างน้อย 10 ตัว ที่แตกต่างกัน ข้อใดต่อไปนี้จะถูกต้องที่สุด
- ไม่มีข้อมูล `data` ชุดใดที่ทำให้ผลลัพธ์เป็นการเรียงจากน้อยไปมาก
 - ไม่มีข้อมูล `data` ชุดใดที่ทำให้ผลลัพธ์เป็นการเรียงจากมากไปน้อย
 - มีข้อมูล `data` บางชุดที่ทำให้ผลลัพธ์เรียงจากน้อยไปมาก และมีข้อมูล `data` บางชุดที่ทำให้ผลลัพธ์เรียงจากมากไปน้อย
 - ไม่มีข้อใดถูก

```
void radixSort(vector<int> &data, int d) {
    //d คือจำนวนหลักของ data ที่คำนวณมาถูกต้องแล้ว และ base = 10
    queue<int> q[base];
    for (int k=0; k<d; k++) {
        for (auto &x : data)
            q[getDigit(x,k)].push(x);
        for (int i = 0, j = 0; i < base; i++)
            while (!q[i].empty()) {
                data[j++] = q[i].front(); q[i].pop();
            }
    }
}
```

ในข้อต่าง ๆ ต่อไปนี้เป็น การเขียน code คะแนนที่ได้จะแปรผันตามความถูกต้อง และ ประสิทธิภาพในการทำงาน โดยเฉพาะการเลือก data structure ในการใช้งานที่ถูกต้อง

8. (10 คะแนน) จงเขียนฟังก์ชัน `void reverse(std::queue<int> &q)` ซึ่งจะทำการย้อนกลับลำดับของข้อมูลใน `q` อย่างไรก็ตาม ในฟังก์ชันนี้ ห้ามสร้างตัวแปรประเภทใด ๆ นอกเหนือไปจาก `set`, `pair`, `int`, `iterator` ของ `set`

```
void reverse(std::queue<int> &q) {
```

```
}
```

9. (10 คะแนน) สมมติให้มี `vector<vector<string>> songs` ซึ่งเก็บข้อมูลเพลงในโทรศัพท์ โดยที่ `songs[i]` เก็บข้อมูลเพลง 1 เพลงในรูปแบบ {ชื่อเพลง, ชื่อคนร้อง, ชื่ออัลบั้ม} (กล่าวคือ `songs[i][2]` จะเก็บ ชื่ออัลบั้มเสมอสำหรับทุก ๆ ค่า `i` เป็นต้น) จงเขียนฟังก์ชัน `void sort_song(vector<vector<string>> &v)` ฟังก์ชันนี้เรียงเพลงใน `v` ตาม คนร้อง อัลบั้ม และชื่อเพลง ตามลำดับ ตารางต่อไปนี้จะแสดงตัวอย่างของ `vector v` ก่อนและหลังเรียกฟังก์ชัน

v ก่อนเรียก	v หลังเรียก
<pre>{{"HoshiShong", "Fumi", "Urusei Album"}, {"Aum No Song", "Fumi", "Urusei Album"}, {"Baka Song", "CKo", "BiBiAlbum"}, {"Aoi Song", "CKo", "AiAiAlbum"}, {"BumBum Song", "Fumi", "Annivesary Album"}}</pre>	<pre>{{"Aoi Song", "CKo", "AiAiAlbum"}, {"Baka Song", "CKo", "BiBiAlbum"}, {"BumBum Song", "Fumi", "Annivesary Album"}, {"Aum No Song", "Fumi", "Urusei Album"}, {"HoshiShong", "Fumi", "Urusei Album"}}</pre>

```
void sort_song(vector<vector<string>> &v){
```

```
}
```

10. (5 คะแนน) กำหนดให้ `map<int, float> m` นั้นเก็บข้อมูลของรหัสสินค้า (เป็น `int`) และราคา (เป็น `float` ที่เป็นบวกเสมอ) ในร้านค้าออนไลน์แห่งหนึ่ง จงเขียนฟังก์ชัน `int max_price` ที่คืนค่ารหัสสินค้าที่มีราคามากที่สุด ที่มีรหัสสินค้ามากกว่า `a` และมีรหัสสินค้าน้อยกว่า `b` ในกรณีที่ไม่มีรหัสสินค้าตรงตามเงื่อนไขดังกล่าว ให้คืนค่า `-1` และในกรณีที่ไม่มีรหัสสินค้ามากกว่า 1 สินค้าที่ตรงตามเงื่อนไขดังกล่าว ให้คืนค่ารหัสสินค้าที่มากที่สุดที่ตรงตามเงื่อนไข (ในข้อนี้ ไม่รับประกันว่า `a` จะต้อง `<= b` หรือไม่)

```
int max_price(std::map<int, float> &m, int a, int b) {

}
}
```

11. (10 คะแนน) เกมออนไลน์เกมหนึ่ง เป็นที่นิยมเป็นอย่างมาก และต้องการจัด “ตารางลำดับคะแนน” ของผู้เล่น ผู้เล่นแต่ละคนจะเล่นเกมและมีคะแนนที่สะสมไว้เป็นของตัวเอง เพื่อส่งเสริมให้ผู้เล่นชิงกับเพื่อน ๆ ในกลุ่มได้ง่าย ๆ เกมนี้จึงอนุญาตให้มีการสร้าง ตารางลำดับคะแนน ได้หลายอัน โดยที่แต่ละอันอาจจะมีรายการผู้เล่นไม่เหมือนกันก็ได้

ผู้เล่นแต่ละคนในเกมนี้จะเริ่มต้นด้วยคะแนน 0 คะแนน และเมื่อทำการเล่นนี้จะไปเรื่อย ๆ คะแนนของผู้เล่นจะมีการเปลี่ยนแปลง โดยที่คะแนนของผู้เล่นแต่ละคนจะเปลี่ยนแปลงไม่เหมือนกันก็ได้ ตารางลำดับคะแนนแต่ละอันจะต้องแสดงผู้เล่นเรียงตามคะแนนจากน้อยไปมาก

กำหนดให้ผู้เล่นแต่ละคนสามารถระบุได้ด้วย หมายเลขผู้เล่น ที่แตกต่างกัน และตารางลำดับคะแนนก็สามารถระบุได้ด้วย หมายเลขตาราง เช่นเดียวกัน โดยหมายเลขผู้เล่นและหมายเลขตารางมีคุณสมบัติคือ เป็นตัวเลขจำนวนเต็มบวกที่เก็บค่าได้ในตัวแปร int (หมายเลขอาจจะไม่เรียงกันและไม่จำเป็นต้องติดกันก็ได้ เช่น อาจจะมีหมายเลข 74, 20, 32848 เป็นต้น และ หมายเลขของตารางอาจจะซ้ำกับหมายเลขของผู้เล่นก็ได้ แต่ไม่มีความเกี่ยวข้องกัน)

จงเขียนคลาส LeaderBoard ซึ่งต้องมีฟังก์ชันต่อไปนี้เป็นอย่างน้อย (นิสิตสามารถเพิ่มเติมฟังก์ชันอื่น ๆ ได้)

- void increase_score(int player_id, int score) เป็นการเพิ่มคะแนนให้กับผู้เล่น player_id ไป score คะแนน หากไม่เคยมีผู้เล่นหมายเลข player_id มาก่อน ให้ถือว่าผู้เล่นคนนั้นเป็นผู้เล่นใหม่ที่มีคะแนนเริ่มต้นเป็น score และผู้เล่นคนนั้นไม่ได้อยู่ในตารางลำดับคะแนนใดเลย
- void join_scoreboard(int player_id, int scoreboard_id) เป็นการเพิ่ม ผู้เล่นหมายเลข player_id เข้าสู่ตารางลำดับคะแนนหมายเลข scoreboard_id (หากไม่เคยมี scoreboard_id นี้มาก่อน ให้สร้างตารางคะแนนใหม่ที่มีผู้เล่น 1 คนคือ player_id และให้ถือว่าตารางคะแนนนี้มีหมายเลขเป็น scoreboard_id และ หากไม่เคยมีผู้เล่นหมายเลข player_id ในระบบมาก่อน ให้ถือว่าผู้เล่นคนนั้นเป็นผู้เล่นใหม่ที่มีคะแนนเริ่มต้นเป็น 0)
- void quit_scoreboard(int player_id, int scoreboard_id) เป็นการลบผู้เล่นหมายเลข player_id ออกจากตารางลำดับคะแนนหมายเลข scoreboard_id (หากไม่เคยมี scoreboard_id นี้หรือ player_id นี้มาก่อน ฟังก์ชันนี้จะต้องไม่ทำการใด ๆ)
- vector<int> get_rank(int scoreboard_id, int start, int stop) ฟังก์ชันนี้จะต้องคืนรายการของหมายเลขของผู้เล่นในตารางลำดับคะแนนหมายเลข scoreboard_id ที่มีคะแนนปัจจุบัน (คะแนน ณ ขณะที่เราเรียกฟังก์ชันนี้) อยู่ในช่วงตั้งแต่ start จนถึง stop เรียงตามลำดับคะแนน (เช่น ถ้าตารางคะแนนนี้มีผู้เล่นซึ่งมีคะแนนปัจจุบันเป็น 10,5,2,1,7,4,9 แล้ว start มีค่าเป็น 3 และ stop มีค่าเป็น 7 จะต้องคืนหมายเลขของผู้เล่นที่ได้คะแนนเป็น 4, 5, และ 7 ตามลำดับ) รับประกันว่าฟังก์ชันนี้จะถูกเรียกโดยที่ 0 <= start <= stop แต่ไม่รับประกันว่า scoreboard_id จะเป็นค่าที่เคยพบมาก่อน (ในกรณีนี้ให้คืน vector ว่าง) ถ้าหากมีผู้เล่นที่มีคะแนนเท่ากันในช่วงคะแนนที่ต้องการ สามารถรายงานผู้เล่นที่คะแนนเท่ากันในลำดับใดก็ได้ที่ตรงกับที่รายการนั้นยังเรียงตามลำดับคะแนนจากน้อยไปมาก

ฟังก์ชันต่าง ๆ เหล่านี้จะถูกเรียกในลำดับใด ๆ ก็ได้ และสามารถถูกเรียกได้หลายครั้งด้วยค่าที่อาจจะเหมือนหรือแตกต่างกันก็ได้ ในการออกแบบคลาสนี้ขอให้ถือสมมติฐานดังนี้

- ผู้เล่นแต่ละคนจะอยู่ในตารางลำดับคะแนนไม่เกิน 10 ตาราง จำนวนผู้เล่นมีเยอะแยะมากมาย และจำนวนผู้เล่นในตารางลำดับคะแนนอาจจะมีเยอะแยะมากมาย
 - ฟังก์ชัน get_rank และ increase_score ถูกเรียกใช้งาน บ่อยกว่า join และ quit มากมายมหาศาล
- จงตอบคำถามต่อไปนี้

11.1. จงระบุชื่อและประเภทของ Data Member ที่ใช้ในคลาสนี้ พร้อมทั้งระบุวัตถุประสงค์ของ Data Member ดังกล่าว

--

11.2. จงเขียนคลาสดังกล่าว (ให้ถือว่าสามารถใช้ Data Structure และฟังก์ชันต่าง ๆ ของ C++ ได้โดยไม่จำกัด)

--

STL Reference

Common (All classes support these two capacity functions)

Capacity	<pre>size_t size(); // return the number of items in the structure bool empty(); // return true only when size() == 0</pre>
----------	---

Container Class (All classes in this category support these two iterator functions.)

Iterator	<pre>iterator begin(); // an iterator referring to the first element iterator end(); // an iterator referring to the <i>past-the-end</i> element</pre>
----------	--

vector<T> และ list<T>

Element Access สำหรับ vector	<pre>T& operator[] (size_t n); T& at(int dx);</pre>
Modifier ที่ใช้ได้ทั้ง list และ vector	<pre>void push_back(const T& val); void pop_back(); iterator insert(iterator position, const T& val); iterator insert(iterator position, InputIterator first, InputIterator last); iterator erase(iterator position); iterator erase(iterator first, iterator last); void clear(); void resize(size_t n);</pre>
Modifier ที่ใช้ได้ เฉพาะ list	<pre>void push_front(const T& val); void pop_front(); void remove(const T& val);</pre>

set<T>

Operation	<pre>iterator find (const T& val); size_t count (const T& val);</pre>
Modifier	<pre>pair<iterator,bool> insert (const T& val); void insert (InputIterator first, InputIterator last); iterator erase (iterator position); iterator erase (iterator first, iterator last); size_t erase (const T& val);</pre>

map<KeyT, MappedT>

Element Access	<pre>MappedT& operator[] (const KeyT& k);</pre>
Operation	<pre>iterator find (const KeyT& k); size_t count (const KeyT& k);</pre>
Modifier	<pre>pair<iterator,bool> insert (const pair<KeyT,MappedT>& val); void insert (InputIterator first, InputIterator last); iterator erase (iterator position); iterator erase (iterator first, iterator last); size_t erase (const KeyT& k);</pre>

Container Adapter

These three data structures support the same data modifiers but each has different strategy. These data structures do not support iterator.

Modifier	<pre>void push (const T& val); // add the element void pop(); // remove the element</pre>
----------	---

	queue<T>	stack<T> and priority_queue<T, ContainerT = vector<T>, CompareT = less<T> >
Element Access	<pre>T front(); T back();</pre>	<pre>T top();</pre>

Useful functions

```
iterator find(iterator first, iterator last, const T& val);
void sort(iterator first, iterator last, Compare comp);
void lower_bound(iterator first, iterator last, const T& val);
void upper_bound(iterator first, iterator last, const T& val);
pair<T1,T2> make_pair (T1 x, T2 y);
```