

FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 INTRODUCTIONS TO DATA STRUCTURE
Year II, First Semester, Midterm Examination, Sep 30, 2021 08:30-11:30

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 13 ข้อ ในกระดาษคำถามคำตอบ 10 หน้า
2. **ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ**
 - อนุญาตให้ดูข้อมูลจาก web cppreference.com และ cplusplus.com และ mycourseville.com เท่านั้น
 - Mycourseville.com ใช้เพื่ออ่านโจทย์และส่งคำตอบเท่านั้น ไม่อนุญาตให้เปิดเอกสารอื่นใดในเว็บไซต์ดังกล่าว
3. **ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ**
 - ในการสอบ online สามารถใช้ คอมพิวเตอร์ได้
4. ห้ามการหยิบยื่นสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยื่นให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. **นิตินัยการกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับสัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

ห้ามนิตินัยการกระทำผิดเกี่ยวกับการสอบ และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่า
นิตินัยการกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และ
อาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

* ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ *

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....

วันที่.....

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

1. (10 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 คนติดเชื้อโควิดแต่ละจังหวัด ซึ่งแต่ละจังหวัด จะมีข้อมูลของแต่ละวันที่ (วันเดือนปี) ซึ่งแต่ละวันจะมี (จำนวนคนติดเชื้อ จำนวนคนเสียชีวิต และจำนวนคนฉีดวัคซีน) ของวันนั้น ๆ โดยจะต้องสามารถหาข้อมูลได้อย่างรวดเร็วด้วยการป้อนชื่อจังหวัด และวันที่ ได้

`map<pair<string, string>, tuple<int, int, int>>`

1.2. ต้องการเก็บข้อมูลหูฟังไร้สาย ข้อมูลจะมี รุ่น, ราคา, review score (อาจมีหูฟังที่คะแนนเท่ากัน), คุณสมบัติ (ซึ่งแต่ละหูฟังมีได้มากกว่าหนึ่งอย่าง เช่น sleep, noise cancel, shape memory ฯลฯ ชนิดของคุณสมบัตินี้มีจำนวนมาก (แต่ไม่เท่าจำนวนรุ่น) และหูฟังหนึ่งๆจะไม่มีคุณสมบัติเกิน 2 อย่าง) จะต้องหาหูฟังได้ด้วยการระบุคุณสมบัติ (แต่ระบุได้เพียงอย่างเดียวต่อการหาหนึ่งครั้ง) และโปรแกรมจะลิสต์หูฟังที่มีคุณลักษณะนั้นมาให้ทั้งหมดได้ พยายามให้มีการเก็บข้อมูลซ้ำซ้อนน้อยที่สุด

`vector<`

1.3. จากข้อ 1.2 ถ้าอนุญาตให้มีการเก็บข้อมูลซ้ำซ้อนได้ จะใช้โครงสร้างข้อมูลอย่างไร ถึงน่าจะทำงานได้เร็ว

`map<string, vector<tuple<string, int, int>>>`

1.4. ต้องการที่เก็บข้อมูล ร้านขายสัตว์เลี้ยง ที่สามารถค้นหาได้ด้วย ชนิดของสัตว์และชื่อสถานที่ใกล้เคียง (ซึ่งสามารถเจอมากกว่าหนึ่งร้าน) ตัวร้านเอง มีข้อมูล คือ ชื่อร้าน และ คำบรรยายร้าน

`map<pair<string, string>, vector<pair<string, string>>>`

1.5. ต้องการเก็บข้อมูลคนรอหน้าร้านอาหารบาร์บีคิวปลาซ่า โดยสิ่งที่ต้องการเก็บคือ หมายเลขที่มาบอกจองที่นั่งและชื่อของคนจอง เพื่อไว้เรียกและตรวจสอบคนเข้าร้าน

`map<string, string>`

2. (4 คะแนน) จากโค้ดด้านล่างนี้ จงเขียนผลลัพธ์ที่แสดงออกทางหน้าจอ

```
std::priority_queue<int> pq;
std::vector<int> v = {3,1,6,4,3};
for (auto &x : v) pq.push( -x );
while (!pq.empty()) {
    std::cout << -pq.top() << " ";
    pq.pop();
}
```

1 3 3 4 6

3. (4 คะแนน) ให้ `v` เป็น `vector<int>` ที่มีข้อมูลเป็นดังนี้ {7,8,20,1,2,5,6,8,3}

จงเขียนคำสั่งเดียว (มี ; เพียงอันเดียว) ที่จะทำให้ข้อมูลใน `v` เป็นดังในแต่ละข้อต่อไปนี้ (แต่ละข้อไม่เกี่ยวข้องกันให้ถือว่าก่อนเรียกคำสั่งในแต่ละข้อ `v` มีข้อมูลดังข้างบน)

3.1 {7,8,14,20,1,2,5,6,8,3} `v.insert(v.begin()+2, 14);`

3.2 {} `v.erase(v.begin(), v.end());`

3.3 {7,8,20,1,2,5,6,8,3,91} `v.insert(v.end(), 91);`

3.4 {7,8,20,1,5,6,8,3} `v.erase(v.begin()+4);`

4. (6 คะแนน) จงพิจารณาส่วนของโปรแกรมต่อไปนี้ ซึ่งต้องการพิมพ์ข้อมูลทั้งหมดใน CP::queue จากท้าย queue ในส่วนของโปรแกรมนี้ที่มีผิดพลาดอยู่ จงระบุบรรทัดที่ผิดพลาด พร้อมทั้งอธิบายว่าความผิดพลาดดังกล่าวคืออะไร

```

1: template <typename T>
2: class queue { //คลาส queue นี้ทำงานได้ตามปรกติทุกอย่าง มีเพียงฟังก์ชัน print_from_back() ที่เพิ่มเข้ามาเท่านั้น
3: void print_from_back() {
4:     int pos = (mFront + mSize) % mCap;
5:     for (int i = 0; i < mCap; i++) {
6:         std::cout << mData[pos] << std::endl;
7:         pos--;
8:     }
9: }
10:};

```

4 ; ต้องเป็น (mFront + mSize - 1) % mCap

5 ; เปลี่ยน mCap เป็น mSize

6 ; pos เริ่มต้นเป็นลบได้

5. (6 คะแนน) ส่วนของโปรแกรมต่อไปนี้ในแต่ละข้อจะมีการใช้งาน iterator ชื่อว่า it ให้พิจารณาว่า หลังจากจบการทำงานในแต่ละข้อแล้ว ตัวแปร it นั้นเป็นตัวแปรที่ valid อยู่หรือไม่

ข้อ ย่อย	ส่วนของโปรแกรม	เป็น iterator ที่ valid หรือไม่?	ถ้าเป็น ให้ระบุค่า *it ถ้าไม่เป็น ให้ระบุเลขบรรทัด สุดท้ายที่ยังคงเป็น
(1)	<pre> 1: CP::vector<int> v(10); 2: auto it = v.begin(); 3: v.resize(21); </pre>	X	2
(2)	<pre> 1: std::set<int> s; 2: s.insert(1); 3: s.insert(2); 4: auto it = s.lower_bound(2); 5: s.insert(3); </pre>	✓	2
(3)	<pre> 1: CP::vector<int> v(100); 2: for (int i = 0; i < 100; i++) 3: v[i] = 100-i; 4: auto it = v.begin(); 5: std::sort(v.begin(), v.end()); 6: v.erase(v.begin()); </pre>	✓	2

ในข้อต่าง ๆ ต่อไปนี้เป็นการเขียน code คะแนนที่ได้จะแปรผันตามความถูกต้อง และ ประสิทธิภาพ
ในการทำงาน โดยเฉพาะการเลือก data structure ในการใช้งานที่ถูกต้อง

6. (5 คะแนน) กำหนดให้มี vector<int> x อยู่ โดยที่ข้อมูลใน x นั้นเรียงจากน้อยไปมากแล้ว เราต้องการทราบว่าใน x มีค่า y อยู่กี่ค่า (เช่น ให้ x = {1,2,3,4,5,5,10} และ y = 5 คำตอบคือ 3 (เนื่องจากมี 5 อยู่ 3 ตัวใน x) จงเขียนฟังก์ชัน int count(vector<int> &x, int y) เพื่อตอบคำถามดังกล่าว

```

int count(vector<int> &x, int y) {
    auto it = lower_bound(v.begin(), v.end(), y);
    int cnt = 0;
    for (int i = it - v.begin(); i < v.size(); i++) {
        if (v[i] != y) break;
        cnt++;
    }
    return cnt;
}

```

$O(N)$

return (lower_bound(v.begin(), v.end(), y+1) - v.begin()) - (lower_bound(v.begin(), v.end(), y) - v.begin());

$O(\log n)$

7. (10 คะแนน) โจทย์ข้อนี้ต้องการให้นักศึกษาสร้าง class ชื่อว่า CP::simple_set<T> ที่มีความสามารถคล้ายกับ std::set ขึ้นมา โดยคลาส simple_set นี้ได้กำหนด data member มาให้แล้วเป็นประเภท std::vector<T> จำนวน 1 ตัว ห้ามนิสิตทำการเพิ่ม data member ใดๆเพิ่มเติม คลาสนี้จะต้องมีฟังก์ชันต่อไปนี้

- pair<iterator,bool> insert(const T& value) ซึ่งจะทำหน้าที่ใส่ค่า value เข้าไปใน simple_set ก็ต่อเมื่อใน simple_set ของเราไม่มีค่า value อยู่ แต่ถ้าหากใน simple_set มีค่า value อยู่แล้ว ฟังก์ชันนี้จะไม่เปลี่ยนแปลง simple_set ของเรา ฟังก์ชันนี้จะต้องคืนค่า pair ของ iterator และ bool โดยที่ iterator ชี้ไปยังข้อมูลในช่องที่มีค่า value อยู่ และ bool คืนค่า true เมื่อการเรียก insert นี้ทำให้มีการเพิ่มข้อมูลเข้าไปใน simple_set ของเรา
- iterator find(const T&value) ซึ่งจะทำหน้าที่ค้นหาว่าใน simple_set ของเรามีค่า value อยู่หรือไม่ หากมีให้คืน iterator ที่ชี้ไปยังค่าดังกล่าว หากไม่มีให้ชี้ไปยัง end()

นอกจากนี้ iterator ของ simple_set นั้นถูกกำหนดให้เป็นประเภทเดียวกับ iterator ของ std::vector<T> ซึ่ง iterator นี้จะต้องทำงานในลักษณะเดียวกันกับ set กล่าวคือ begin() จะต้องคืนค่า iterator ที่ชี้ไปยังตัวที่มีค่าน้อยที่สุด และการใช้ ++ และ - จะทำการเปลี่ยนค่าของ iterator ในแบบเดียวกับ set ก็คือการเปลี่ยนตัวชี้ไปยังตัวที่มี “ค่า” อยู่ติดกันนั่นเอง (ตัวอย่างเช่น การพิมพ์ข้อมูลจาก iterator ของคลาสนี้จากตัวแรก (begin) ไปยังตัวสุดท้าย (ก่อน end) จะได้ค่าที่เรียงจากน้อยไปมาก)

คลาสนี้ได้มีการกำหนด member function อื่น ๆ มาให้แล้ว ได้แก่ begin(), end() และ constructor รวมถึงมีการกำหนดประเภท iterator ไว้ให้แล้วเช่นกัน ให้นักศึกษาเขียนเพียง 2 ฟังก์ชันข้างบนเพิ่มเติมโดย ต้องไม่ทำการเปลี่ยนแปลงส่วนอื่นใดของคลาส CP::simple_set นี้ และให้ถือว่าคลาส T ที่รับเข้ามาเป็นคลาสที่สามารถเปรียบเทียบได้อยู่แล้ว

```
namespace CP {
    template <typename T>
    class simple_set {
    private:
        std::vector<T> mData; //ห้ามสร้าง data member เพิ่มเติม
    public:
        //iterator ของ simple_set คือ iterator ของ vector<T>
        typedef typename std::vector<T>::iterator iterator;
        iterator begin() { return mData.begin(); }
        iterator end() { return mData.end(); }

        pair<iterator,bool> insert(const T& value) {
            //write your code here
            int i; bool odd = false;
            for(i=0; i<mData.size(); i++){
                if(mData[i] > value){
                    mData.insert(mData.begin()+i, value);
                    odd = true;
                    break;
                }
                else if(mData[i] == value) break;
            }
            return {mData.begin()+i, odd};
        }

        iterator find(const T& value) {
            //write your code here
            int i;
            for(i=0; i<mData.size(); i++){
                if(mData[i] == value) break;
                if(i == mData.size()-1) i++;
            }
            return mData.begin()+i;
        }
    };
}
```

8. (10 คะแนน) ในข้อนี้ให้นิสิตเพิ่มบริการ void moveInsertStack(int k, CP::stack<T>& s, int m) ให้กับ class CP::stack เพื่อทำการย้ายข้อมูล m ตัวบนสุดของกองซ้อน s ไปยังกองซ้อน *this โดยนำไปแทรก ณ ตำแหน่งที่ k นับจากด้านบนของกองซ้อน (k = 0 จะหมายถึงให้วางเหนือข้อมูลทุกตัวของ *this) โดยหาก k มีค่ามากกว่าหรือเท่ากับจำนวนข้อมูลใน *this ก็ให้ไปแทรกได้สุดของกองซ้อน หาก m มากกว่าหรือเท่ากับ s.size() ก็คือให้ย้ายข้อมูลทุกตัว

ตัวอย่างเช่น ให้ s1 เป็น stack ที่มีข้อมูลเป็น [2, 7, 4, 3] โดยด้านบนของกองซ้อนคือตัวซ้ายสุด และ s2 มีข้อมูลเป็น [10, 6, 9] การเรียก s1.moveInsert(1, s2, 2) จะทำให้ s1 มีข้อมูลเป็น [2, 10, 6, 7, 4, 3] และ s2 มีข้อมูลเป็น [9]

โดยในข้อนี้ให้นิสิตแก้ไขเฉพาะ void moveInsertStack(int k, CP::stack<T>& s, int m) ใน stack.h เท่านั้นโดยห้ามเพิ่ม data member หรือ member function ใด ๆ เพิ่มเติม

```
template <typename T>
class stack {
protected:
    T *mData; size_t mCap, mSize, mTop; // ห้ามประกาศ data member เพิ่มเติม
public:
    // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง โดยฟังก์ชันอื่น ๆ มิได้ระบุไว้เพื่อประหยัดหน้ากระดาษ ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
    // ให้เขียนเต็มฟังก์ชันด้านล่างนี้ให้ทำงานให้ถูกต้อง
    void moveInsertStack(int k, stack<T>& s, int m) {
        CP::stack temp;
        for(int i = 0; i < size() - k; i++) temp.push(mData[i]);
        for(int i = s.size() - m; i < s.size(); i++) temp.push(s.mData[i]);
        for(int i = size() - k; i < size(); i++) temp.push(mData[i]);
        *this = temp;
        s.mSize -= m;
    }
};
```

9. (10 คะแนน) อ. โต (อีกแล้ว) กำลังเล่นเกม Magix The Burrowing อยู่ ซึ่งเป็นเกมที่ผู้เล่นจะสะสมการ์ดต่าง ๆ จำนวนหลาย ๆ ใบ ปัจจุบัน อ. โตมีการ์ดอยู่ N ใบ การ์ดแต่ละใบจะประกอบด้วยข้อมูล 3 อย่างดังต่อไปนี้
- ชื่อการ์ด เป็นตัวอักษรภาษาอังกฤษติดกันหลายไม่มีช่องว่างและไม่มีตัวเลข
 - สีของการ์ด การ์ดแต่ละใบจะมีสีได้สี่เดียว โดยสีที่เป็นไปได้คือ แดง เขียว น้ำเงิน ซึ่งระบุด้วยตัวอักษรภาษาอังกฤษ 1 ตัวคือ R, G, B ตามลำดับ
 - ค่าพลังของการ์ดเป็นจำนวนเต็มตั้งแต่ 0 เป็นต้นไป
 - การ์ดที่มีชื่อเหมือนกันจะมีสีและค่าพลังเหมือนกันแน่นอน
- จงเขียนโปรแกรมเพื่อรับข้อมูลการ์ดที่ อ. โตมีทั้งหมด แล้วแสดงผลการ์ดที่ อ. โตจะพิมพ์ข้อมูลเข้าสู่โปรแกรมตามรูปแบบต่อไปนี้
- บรรทัดแรกระบุจำนวนเต็ม N ซึ่งคือจำนวนของการ์ดทั้งหมดที่มี
 - หลังจากนั้นอีก N บรรทัดจะเป็นข้อมูลการ์ดแต่ละใบ บรรทัดละ 1 ใบโดยมีรูปแบบคือ ชื่อการ์ด ตามด้วยสีของการ์ด และตามด้วยค่าพลังของการ์ด
- การแสดงผลให้แสดงผลการ์ดทั้งหมดเรียงตามลำดับต่อไปนี้
- สี (น้ำเงิน ตามด้วย แดง ตามด้วย เขียว)
 - ถ้าสีเหมือนกันให้เรียงตามค่าพลัง จากน้อยไปมาก
 - ถ้าสีและค่าพลังเหมือนกันให้เรียงตามชื่อตามลำดับอักษรภาษาอังกฤษ
- เนื่องจาก อ. โตอาจจะมีการ์ดที่มีข้อมูลเหมือนกันอยู่หลาย ๆ ใบก็เป็นได้ ให้แสดงผลโดยระบุข้อมูลของการ์ด ตามด้วยเครื่องหมาย = และตามด้วยจำนวนของการ์ดนั้นที่มี โดยข้อมูลของการ์ดให้ระบุชื่อของการ์ด ตามด้วย สีของการ์ด และ ตามด้วยพลังของการ์ดตามลำดับ ดังตัวอย่างด้านล่างนี้

ตัวอย่าง input	ตัวอย่าง output
13	Annul B 1 = 1
Mystic G 1	Disappear B 3 = 2
Imp R 6	Dissipate B 3 = 1
Dissipate B 3	Lightning R 1 = 1
Shock R 1	Shock R 1 = 2
Shock R 1	Imp R 6 = 1
Mystic G 1	Mystic G 1 = 2
Disappear B 3	BigHug G 2 = 1
Annul B 1	BigPlay G 2 = 1
Wood G 4	Wood G 4 = 1
Lightning R 1	
Disappear B 3	
BigHug G 2	
BigPlay G 2	

ในข้อนี้มีโค้ดเริ่มต้นสำหรับการอ่านข้อมูลให้แล้ว จงเขียนโปรแกรมต่อโดยใช้โค้ดตั้งต้นดังกล่าว นิสิตสามารถสร้างคลาสเพิ่มเติมได้ และสามารถเรียกใช้ฟังก์ชันหรือคลาสอื่นใด ๆ ได้ (ไม่จำเป็นต้อง include) การแสดงผลใช้ cout, endl ได้เลยไม่ต้องกังวลว่า cout, endl จะซ้ำ (ไม่จำเป็นต้องเรียก sync_with_stdio, ฯลฯ)

```
int main() {
    // you can add more data here
    vector<Card> v;
    map<string, int> m;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        string name, color;
        int power;
        cin >> name >> color >> power;
```



//add your code here

```
if (!m.count(name)) v.push_back(Card(color, power, name));
m[name]++;
```

}

```
sort(v.begin(), v.end());
```

```
for (auto e : v)
```

```
cout << e.getName() << " " << e.getColor()
```

```
<< " " << e.getPower() << " = " << m[e.getName()]
<< endl;
```

}

```
class Card {
protected:
    char color;
    int power;
    string name;
public:
    Card(char color, int power, string name) {
        color(color), power(power), name(name);
    }
    bool operator<(const Card &other) const {
        if (color == 'G' and other.color == 'R') return false;
        else if (color == 'R' and other.color == 'G') return true;
        else if (color != other.color) return color < other.color;
        else if (power != other.power) return power < other.power;
        return name < other.name;
    }
    string getName() { return name; }
    string getColor() { return color; }
    int getPower() { return power; }
}
```

10. (10 คะแนน) จงเขียนฟังก์ชัน `vector<int> moving_median(vector<int> &data, int w)` ซึ่งคำนวณ “ค่ากลางเคลื่อนที่” ของ data โดยกำหนดให้ “ค่ากลางตำแหน่ง i” คือ ค่ากลางของข้อมูล data ในช่วงหมายเลข [i] จนถึงหมายเลข [i+w] และกำหนดให้ “ค่ากลาง” (median) ของข้อมูลของ data ในช่วงหมายเลข [a] ถึง หมายเลข [b] นั้นคือข้อมูลที่มีค่ามากที่สุดเป็นลำดับที่ $1 + (b-a)/2$ ของข้อมูลของ data ตั้งแต่ช่วงหมายเลข [a] ถึงหมายเลข [b] ที่เรียงแล้ว ตัวอย่างเช่น สมมติให้ data มีค่าเป็น {1,3,2,4,5,1,2} และ w มีค่าเป็น 2 แล้ว ค่ากลางตำแหน่ง 0 จะมีค่าเป็น 2 (คำนวณจากข้อมูล data[0] .. data[2] ซึ่งคือ {1,3,2}), ค่ากลางตำแหน่ง 1 จะมีค่าเป็น 3 (คำนวณจาก data[1]..data[3] ซึ่งคือ {3,2,4}) และ ค่ากลางตำแหน่ง 2 จะมีค่าเป็น 4 (คำนวณจากข้อมูล {2,4,5})

ฟังก์ชันนี้จะต้องคืนค่า `vector<int>` ที่มีจำนวนข้อมูลเท่ากับ `data.size()-w` และ ข้อมูลในตำแหน่งที่ i ใน `vector<int>` ที่คืนมานั้นก็คือ “ค่ากลางของตำแหน่ง i” นั่นเอง โดย `vector<int>` ที่คืนมานี้จะเรียกว่า “ค่ากลางเคลื่อนที่”

รับประกันว่า data นั้นมีขนาดไม่น้อยกว่า 1 และ มีค่าไม่ซ้ำกันเลย และรับประกันว่า w เป็นจำนวนคู่ที่มีค่าไม่น้อยกว่า 1 และ ไม่มากกว่า `data.size()`

หากกำหนดให้ data = {1,3,2,4,5,1,2} และ w เป็น 2 นั้น ผลที่ได้จากการเรียกฟังก์ชันนี้คือ {2,3,4,4,2}

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

--	--	--

```
vector<int> moving_median(vector<int> &data, int w) {
```

```
}
```

11. (10 คะแนน) จงเขียนฟังก์ชัน `int furthest(vector<int> &v)` เพื่อหาว่ามีคู่ตำแหน่ง a, b ใด ๆ ที่ $v[a]$ และ $v[b]$ มีค่าเท่ากัน และค่า $b-a$ มีค่ามากที่สุด (กล่าวคือ เราต้องการหาช่องสองช่องที่มีค่าเหมือนกัน แต่หมายเลขช่องอยู่ห่างกันมากที่สุด) โดยให้คืนค่าเป็นค่าของ $b-a$ โจทย์ข้อนี้รับประกันว่าใน v มีอย่างน้อย 2 ช่องที่มีค่าเท่ากัน

```
int furthest(vector<int> &v) {
```

n log n

```
};
```

12. (10 คะแนน) “min of max” อ. แด่กำลังเล่นเกมเกมหนึ่ง ในเกมนี้ผู้เล่นจะต้องจัด “ทีม” เพื่อไปแข่งขัน ทีมประกอบด้วยตัวละครหลาย ๆ ตัว โดยตัวละครแต่ละตัวจะมี “ค่าพลัง” และ “ค่าประเภท” กำกับอยู่ (กำหนดให้ค่าประเภทนี้มีค่าตั้งแต่ 0 ถึง $m-1$ เท่านั้น) ทีมจะต้องประกอบด้วยตัวละครจำนวน m ประเภทพอดี โดยจะมีประเภทละกี่ตัวก็ได้ เกมนี้กำหนดให้ “ความแข็งแกร่งของทีม” คือค่าน้อยสุดของ “พลังประเภท i ” สำหรับค่า i ตั้งแต่ 0 ถึง $m-1$ ของทีมนี้ และให้ “พลังประเภท i ” ของทีมคือค่าพลังที่มากที่สุดของตัวละครประเภท i ในทีมนี้

อ. แด่ เริ่มต้นด้วยตัวละคร m ตัว โดยแต่ละตัวจะมีค่าพลังเป็น 1 และมีค่าประเภทตั้งแต่ 0 ถึง $m-1$ พอดี ทำให้ทีมของ อ. แด่มีค่าพลังเป็น 1 ในตอนเริ่มต้น อ. แด่ค่อย ๆ ได้รับตัวละครเพิ่มเข้ามาในทีมนี้ทีละตัว รวมได้มาเพิ่มทั้งหมด n ตัว เราต้องการทราบ “ความแข็งแกร่งของทีม” ของ อ. แด่ หลังจากการได้รับตัวละครแต่ละตัวตามลำดับ

จงเขียนฟังก์ชัน `vector<int> team_power(vector<int> &power, vector<int> &type, int m)` ซึ่งรับข้อมูลการซื้อตัวละคร โดยที่ `power[i]` คือค่าพลังของตัวละครที่ได้มาในลำดับที่ i และ `type[i]` คือค่าประเภทของตัวละครที่ได้มาในลำดับที่ i โดยฟังก์ชันนี้จะต้องคืน `vector<int>` ที่มีข้อมูล n ตัวโดยที่ข้อมูลในช่องหมายเลข i ของ `vector` นี้คือ “ความแข็งแกร่งของทีม” หลังจากที่ได้ตัวละครลำดับที่ i มาอยู่ในทีม

ตัวอย่างผลการเรียก `team_power({1, 10, 5, 15, 12}, {0, 0, 1, 1, 0}, 2)` คือ `{1,1,5,10,12}`

```
vector<int> team_power(vector<int> &power, vector<int> &type, int m) {
```

pq

```
}
```

13. (10 คะแนน) เราเที่ยวด้วยกัน!!! รัฐบาลต้องการสนับสนุนการท่องเที่ยว จึงได้จัดทำโครงการคืนเงินค่าตัวเครื่องบินสำหรับการไปพักผ่อนยังโรงแรมต่าง ๆ โครงการนี้เกี่ยวข้องกับคน โรงแรม และเที่ยวบิน โดยมีกฎดังต่อไปนี้

- ให้คนแต่ละคนระบุได้ด้วย “หมายเลขประจำตัวประชาชน” เป็นตัวเลขจำนวนเต็มแบบ `int`
- ให้โรงแรมแต่ละโรงแรมระบุได้ด้วย “ชื่อโรงแรม” เป็น `string`

*people
int*

hotel string → some area

- some area*
flight string, int
area int
- ให้เที่ยวบินถูกระบุได้ด้วย “รหัส” พร้อมกับ “วันที่” โดย รหัสเป็น string และ วันที่ระบุด้วย int
 - เพื่อความสะดวก วันที่ในระบบนี้จะแบบ int โดยนับเป็นจำนวนวันที่ผ่านมาตั้งแต่ 1 ม.ค. 2563
 - ตัวเครื่องบินและโรงแรมจะเกี่ยวข้องกับ “พื้นที่” ให้สมมติว่าในโครงการนี้มีพื้นที่ที่เป็นไปได้ทั้งหมด n พื้นที่ พื้นที่แต่ละพื้นที่ระบุได้ด้วยจำนวนเต็มแบบ int
 - รัฐบาลกำหนดไว้ว่าโรงแรมแต่ละแห่งนั้นรองรับพื้นที่ใดบ้าง
 - รัฐบาลจะกำหนดไว้ว่าเที่ยวบินในแต่ละ “รหัส” นั้นรองรับพื้นที่ใดบ้าง (โดยถือว่าเที่ยวบินที่ใช้รหัสเดียวกันรองรับพื้นที่เหมือนกันเสมอ ไม่ว่าจะเดินทางในวันใดก็ตาม)
 - นักท่องเที่ยวจะได้เงินค่าตัวเครื่องบินคืนก็ต่อเมื่อพักผ่อนในโรงแรมในวันที่ x และมีการเดินทางด้วยเที่ยวบินในระหว่างวันที่ $x-w$ ถึงวันที่ $x+w$ และ รายการของพื้นที่ที่รองรับโดยเที่ยวบินดังกล่าว และ รายการพื้นที่ที่รองรับโดยโรงแรมดังกล่าวมีอย่างน้อย 1 พื้นที่เหมือนกัน
 - ตัวอย่างเช่น โรงแรม A รองรับพื้นที่ “เชียงใหม่” และ “เชียงราย” ส่วนเที่ยวบิน WD115 รองรับพื้นที่ “เชียงราย” และ “แม่ฮ่องสอน” หากนักท่องเที่ยวพักโรงแรม A ในวันที่ 5 โดยเดินทางด้วยเที่ยวบิน WE115 ในวันที่ 4 (สมมติให้ค่า w เป็น 1) นักท่องเที่ยวดังกล่าวก็สามารถขอเงินคืนได้
 - แต่ถ้านักท่องเที่ยวเดินทางด้วยเที่ยวบิน FD332 ซึ่งรองรับพื้นที่ “แม่ฮ่องสอน” และ “ลำพูน” ก็ไม่สามารถขอเงินคืนได้ (เนื่องจากไม่มีพื้นที่อย่างน้อย 1 พื้นที่ที่เหมือนกันระหว่างการพักผ่อนกับการเดินทาง) หรือหากนักท่องเที่ยวเดินทางด้วย WE115 แต่เดินทางในวันที่ 3 หรือ 7 ก็ไม่สามารถขอเงินคืนได้เช่นกัน (เนื่องจากวันที่เดินทางไม่อยู่ในช่วง $x-w$ ถึง $x+w$ ของการพักผ่อน)

เราต้องการสร้างระบบสำหรับการจัดการการเงิน โดยสร้างคลาส Travel ซึ่งทำหน้าที่ดังกล่าว โดยคลาสนี้จะต้องรับข้อมูลพื้นที่ที่รองรับของโรงแรมต่าง ๆ และเที่ยวบินต่าง ๆ พร้อมทั้งรับข้อมูลว่า เที่ยวบินแต่ละเที่ยวบินมีใครใช้บริการบ้าง และ โรงแรมแต่ละโรงแรมมีใครเข้าพักในวันใดบ้าง คลาสนี้จะทำหน้าที่คำนวณว่าค่าขอคืนเงินแต่ละค่าขอคืนนั้นสามารถทำได้หรือไม่

คลาส Travel ต้องมีฟังก์ชันต่อไปนี้เป็นอย่างน้อย (นิสิตสามารถเขียนฟังก์ชันอื่นเพิ่มเติมได้)

- `Travel(vector<pair<string,vector<int>>> &hotel, vector<pair<string,vector<int>>> flight,int w)` เป็น constructor ซึ่งรับข้อมูล hotels และ flights ซึ่งระบุโรงแรมแต่ละโรงแรม และ เที่ยวบินแต่ละเที่ยวบินรองรับพื้นที่ใดบ้าง โดยให้ `hotels[i].first` คือชื่อของโรงแรม และ `hotels[i].second` คือรายการของพื้นที่ที่โรงแรมนั้นรองรับ และ `flights[i].first` คือชื่อของเที่ยวบิน และ `flights[i].second` คือรายการของพื้นที่ที่เที่ยวบินนั้นรองรับ และ w คือจำนวนวัน
- `void add_stay(string hotel, int date, vector<int> tourist)` เป็นฟังก์ชันที่ใช้ระบุโรงแรมชื่อ hotel ในวันที่ date นั้นมีผู้เข้าพักซึ่งมีเลขประจำตัวตามที่ระบุในตัวแปร tourist
- `void add_passenger(string flight, int date, vector<int> tourist)` เป็นฟังก์ชันที่ใช้ระบุเที่ยวบินรหัส flight ในวันที่ date นั้นมีผู้เดินทางซึ่งมีเลขประจำตัวตามที่ระบุในตัวแปร tourist
- `bool can_get_refund(int id, string flight, int date)` เป็นฟังก์ชันสำหรับตรวจสอบว่าคนที่หมายเลขประจำตัวเป็น id นั้นสามารถขอคืนเงินค่าตัวสำหรับเที่ยวบิน flight ในวันที่ date ได้หรือไม่
 - ฟังก์ชันนี้อาจจะถูกเรียกโดยค่า id, flight หรือ date ใด ๆ ก็ได้ รวมถึงค่าที่ไม่เคยพบมาก่อนในระบบก็เป็นได้
 - ฟังก์ชันจะต้องคืนค่า true ก็ต่อเมื่อ นักท่องเที่ยวสามารถได้รับเงินคืนตามกฎหมายข้างต้นเท่านั้น
 - ให้คืนค่า false ในกรณีอื่น ๆ ทั้งหมด

จงตอบคำถามต่อไปนี้

13.1. จงระบุชื่อและประเภทของ Data Member ที่ใช้ในคลาสนี้ พร้อมทั้งระบุวัตถุประสงค์ของ Data Member ดังกล่าว

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

STL Reference

Common (All classes support these two capacity functions)

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

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

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

vector<ValueT> และ list<ValueT>

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

set<ValueT>

Operation	<code>iterator find (const ValueT& val);</code> <code>size_t count (const ValueT& val);</code>
-----------	---

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

--	--	--

Modifier	pair<iterator,bool> insert (const ValueT& val); void insert (InputIterator first, InputIterator last); iterator erase (iterator position); <u>iterator erase (iterator first, iterator last);</u> size_t erase (const ValueT& val);
----------	---

map<KeyT, MappedT>

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

Container Adapter

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

Modifier	void push (const ValueT& val); // add the element void pop(); // remove the element
----------	--

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

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);

```