FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 INTRODUCTIONS TO DATA STRUCTURE
Year II, First Semester, Midterm Examination, Oct 8, 2020 08:30-11:30

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

<u>หมายเหตุ</u>

1. ข้อสอบมีทั้งหมด 12 ข้อ ในกระดาษคำถามคำตอบ 8 หน้า
2. **ไม่อนุญาตให้นำตำราและเอกสารใดๆ เข้าในห้องสอบ**
3. **ไม่อนุญาตให้ใช้เครื่องคำนวณใดๆ**
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่เจ้าหน้าที่ควบคุมการสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบและสมุดคำตอบออกจากห้องสอบ
6. ผู้เข้าสอบสามารถออกจากห้องสอบได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. **นิสิตกระทำผิดเกี่ยวกับการสอบ ตามข้อบังคับจุฬาลงกรณ์มหาวิทยาลัย มีโทษ คือ พ้นสภาพการเป็นนิสิต หรือ ได้รับสัญลักษณ์ F ในรายวิชาที่กระทำผิด และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้**

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

*** ร่วมรณรงค์การไม่กระทำผิดและไม่ทุจริตการสอบที่คณะวิศวกรรมศาสตร์ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือ หรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....สมชาย สาสกุล.....
วันที่.....30/2/2564.....

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

1. (6 คะแนน) ตอบคำถามต่อไปนี้สั้น ๆ ว่า แต่ละปัญหาต้องมีที่เก็บข้อมูลประเภทใด

1.0 ต้องการเก็บรายชื่อนิสิตคณะวิศวฯทุก ๆ รุ่น แต่ละรุ่นมีหมายเลขรุ่นกำกับ เพื่อเขียนเมท็อด **`int getClassID(string name)`** ที่คืนหมายเลขรุ่นของคนชื่อ **name**

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

1.1 ต้องการที่เก็บข้อมูล ร้านอาหาร สำหรับ application แผนที่ ที่สามารถให้ค้นหาได้ด้วย ชื่อสถานที่ (สถานที่เดียวสามารถให้เราเจอมากกว่าหนึ่งร้าน) ร้านหนึ่งร้านจะมีข้อมูลคือ ชื่อร้าน และประเภทอาหารที่ขาย

key string vector string string pair

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

1.2 ต้องการเก็บข้อมูลหนังสือการ์ตูนที่ซื้อมาทั้งหมด (เก็บว่าซื้อเล่มเดิมซ้ำมากี่เล่มด้วย) หนังสือการ์ตูนแต่ละเล่มประกอบด้วย ชื่อเรื่อง หมายเลขเล่ม และเนื้อเรื่องย่อของเล่มนั้น โดยต้องสืบค้นได้จากการป้อนชื่อเรื่องและหมายเลขเล่ม

int string pair map

ชื่อ หมายเลข # ครั้งที่ซื้อ เรื่องย่อ

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

1.3 ต้องการที่เก็บข้อมูลคนที่เข้ามาอยู่ในตึก และเวลาที่เข้าตึก โดยแต่ละคน (ใช้เลขประจำตัวในการ identify) จะต้องถูกบันทึกเวลา (ในรูปของ string) ในการเข้าตึก ไว้ทั้งหมด (คนหนึ่งอาจจะเข้าตึกหลายครั้ง) string

∵ ไม่ได้บอกให้หา

เลขประจำตัว → ใช้ string เพราะเก็บได้หลายหลักมากๆ

เวลาเข้า

∴ ใช้ `vector<pair<string, string>>`

2. (4 คะแนน) กำหนดให้มี priority queue ดังนี้ priority_queue<int> p; ถามว่า หลังจากรันโค้ดด้านล่างนี้แล้ว p.size() เป็นเท่าไหร่ และอะไรคือ p.top()

priority queue → เอาก่อน มาก → น้อย

```
p.push(20);   20
p.push(30);   30 20
p.push(5);    30 20 5
p.pop();      20 5
p.push(10)    20 10 5
p.push(40);   40 20 10 5
p.push(35);   40 35 20 10 5
p.push(15);   40 35 20 15 10 5
p.pop();      35 20 15 10 5
```

ผลของ p.top() คือ 35

ผลของ p.size() คือ 5

3. (4 คะแนน) ในแต่ละข้อย่อยต่อไปนี้ หากเรากำหนดให้ n มีค่า 1,000,000 จงระบุว่า code ในชุด A หรือ B ที่จะทำงานเสร็จก่อนกัน โดยให้ระบุว่า A หรือ B หรือ เลือกที่จะไม่ตอบก็ได้ หากตอบถูก จะได้คะแนนข้อละ 1 คะแนน หากตอบผิด จะได้คะแนน -0.5 คะแนน แต่ถ้าหากไม่ตอบ จะได้ 0 คะแนน

ข้อย่อย	Code A	Code B	คำตอบ
(1)	`vector<int> v(10);` `while (n--) v.size();` O(n)	`vector<int> v(10);` O(n) + resizing `while (n--) v.push_back(1);`	A
(2)	`map<int,int> m;` create + assign `for (int i = 0;i < n;i++) m[i] = i;`	`map<int,int> m;` just assign `for (int i = 0;i < n;i++) m[1] = i;`	B

(3)	`set<int> s;`
	`for (int i = 0;i < n;i++)`
	`s.insert(i)` → insertion + rebalancing
		`priority_queue<int> pq;`
		`for (int i = 0;i < n;i++) pq.push(i)`
		↳ Just insertion
			B

(4)	`queue<int> q;`
	`for (int i = 0;i < n;i++) {`
	`  q.push(i);`
	`  q.pop();`
	`}` faster, no resizing since mCap is always 1
		`stack<int> s;`
		`for (int i = 0;i < n;i++) s.push(i);`
		`for (int i = 0;i < n;i++) s.pop();`
		resize a lot
			A

4. (6 คะแนน) จงพิจารณาส่วนของโปรแกรมต่อไปนี้ ซึ่งเป็นการใช้งาน CP::queue<int> หากเราต้องการให้ mFront, mSize และ mCap มีค่าที่กำหนดให้ เราต้องทำให้ตัวแปร a, b และ c มีค่าเท่าไร?

```
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);
```

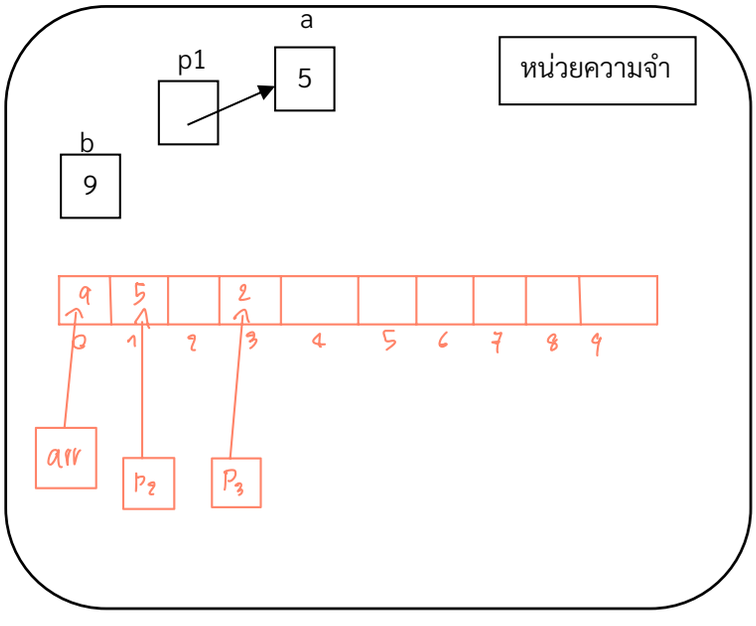
ข้อย่อย	mSize	mCap	mFront	a	b	c
(1)	5	8	5	5	5	5
(2)	16	16	15	15 16	15 15	16 15
(3)	0	32	0	17	17	0

5. (6 คะแนน) จากส่วนของโปรแกรมต่อไปนี้ ให้ระบุว่า หลังจากจบการทำงานแล้ว iterator it ซึ่งเป็น iterator ของ CP::vector นั้น ยังเป็น iterator ที่ชี้ไปยังตำแหน่งที่อยู่ระหว่าง v.begin() จนถึง v.end()-1 หรือไม่ หากเป็น ให้ระบุค่าของ *it และหากไม่เป็น ให้ระบุหมายเลขบรรทัดสุดท้ายที่เมื่อจบการทำงานของบรรทัดนั้นแล้ว it ยังคงเป็น iterator ที่ชี้ไปยังตำแหน่งทีอยู่ระหว่าง v.begin() กับ v.end()-1 อยู่

ข้อย่อย	ส่วนของโปรแกรม	เป็น iterator ที่ชี้ไปยังตำแหน่งระหว่าง v.begin() ถึง v.end()-1 หรือไม่?	ถ้าเป็น ให้ระบุค่า *it ถ้าไม่เป็น ให้ระบุเลขบรรทัดสุดท้ายที่ยังคงเป็น
(1)	1: CP::vector<int> v(10); 2: auto it = v.begin()+9; 3: it++;	No	2
(2)	1: CP::vector<int> v; 2: auto it = v.begin(); 3: for (int i = 0;i < 30;i++){ 4: if (i == 16) 5: it = v.begin(); 6: v.insert(v.begin(),1); 7: }	Yes	1
(3)	1: CP::vector<int> v(100); 2: for (int i = 0;i < 100;i++) 3: v[i] = i; 4: auto it = v.begin() + 42; 5: for (int i = 10;i < 30;i++) 6: v.erase(v.begin());	Yes	62

6. (5 คะแนน) จาก Code ด้านล่างต่อไปนี้ จงวาดรูปแสดงถึงตัวแปรและค่าต่าง ๆ ของตัวแปรในหน่วยความจำของคอมพิวเตอร์หลังจากจบการทำงานในบรรทัดที่ 11 แล้ว (สำหรับตัวแปรประเภท pointer สามารถตอบค่าของตัวแปรโดยการลากลูกศรชี้แทนได้) โดยผลจากการทำงานของบรรทัดที่ 1 – 3 ได้เขียนเป็นตัวอย่างไว้แล้ว

```
1:  int a = 5;
2:  int b = 9;
3:  int *p1 = &a;
4:
5:  int *arr = new int[10]();
6:  int *p2, *p3;
7:  p2 = arr + 1;
8:  p3 = arr + 3;
9:  *p3 = p3 - p2;
10: *p2 = *p1;
11: *(arr) = b;
```



7. (5 คะแนน) กำหนดให้เรามี ตัวแปร m ซึ่งประกาศไว้ดังนี้ map<int, queue<priority_queue<int>>> m จงเขียนส่วนของโปรแกรมที่ทำการพิมพ์ข้อมูลประเภท int ทั้งหมดใน m ออกมา (รวม key ของ map ด้วย) และอนุญาตให้แก้ไข m ได้

```
for (auto e:m){
    cout << e.first;
    while (e.second.size()){
        while(e.second.front().size()){
            cout << e.second.front().top();
            e.second.front().pop();
        e.second.pop();
    }
}
```

8. (10 คะแนน) จงเพิ่มฟังก์ชันเพิ่มเติมให้กับคลาส CP::vector โดยเพิ่มฟังก์ชัน void unique() ซึ่งจะการลบข้อมูลใน vector ที่อยู่ติดกันและซ้ำกันนั้นให้เหลือเพียงตัวเดียว (กล่าวคือไม่มีข้อมูลที่อยู่ติดกันที่มีค่าซ้ำกันเลย) โดยที่ลำดับของข้อมูลยังเหมือนเดิมอยู่ ตัวอย่างเช่น สมมติให้ v = {1,2,2,4,4,1,1,1,1,2,9,8,9} การเรียก v.unique() จะทำให้ v กลายเป็น {1,2,4,1,2,9,8,9}

```
template <typename T>
class vector {
  protected:
    T *mData; size_t mCap, mSize;   // ห้ามประกาศ data member เพิ่มเติม
  public:
    // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง โดยฟังก์ชันอื่น ๆ มิได้ระบุไว้เพื่อประหยัดหน้ากระดาษ ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
    // ให้เขียนเติมฟังก์ชันด้านล่างนี้ ให้ทำงานให้ถูกต้อง
    void unique() {
      if(mSize==0) return;
      T *arr = new T[mCap];
      size_t nSize = 1;
      arr[0] = mData[0];
      for(int i=1;i<mSize;i++){
        if(mData[i]!=mData[i-1]){
          arr[nSize]=mData[i];
          nSize++;
        }
      }
      delete [] mData;
      mData = arr;
      mSize = nSize
    }
};
```

9. (10 คะแนน) มีการ์ดเกมหนึ่ง ซึ่งประกอบด้วยการ์ดจำนวนหลาย ๆ การ์ด การ์ดแต่ละใบเป็นข้อมูลของมอนสเตอร์ในเกม โดยแต่ละใบมีข้อมูลได้แก่ ชื่อการ์ด, พลังโจมตี และ พลังป้องกัน เราต้องการเขียนโปรแกรมเพื่อช่วยในการเล่นเกมดังกล่าว กำหนดให้มีส่วนของโปรแกรมต่อไปนี้เป็นตัวอย่างการเก็บค่าการ์ดต่าง ๆ เข้าไปไว้ใน vector

```
Monster m1 = { "Blue Eyes", {3000, 2500} }; // name = “Blue Eyes” พลังโจมตีคือ 3000, พลังป้องกันคือ 2500
Monster m2 = { "Stardust Dragon", {2500, 2000} };
Monster m3 = { "Dark Magician", {2500, 2100} };
Monster m4 = { "Ancient Gear Golem", {3000, 3000} };
Monster m5 = { "Ancient Gear Golari", {3000, 3000} };
vector<Monster> v;
v.push_back(m1);
v.push_back(m2);
v.push_back(m3);
v.push_back(m4);
v.push_back(m5);
```

จงตอบคำถามต่อไปนี้

a) (1 คะแนน) ประเภทข้อมูล Monster นั้น ควรจะเป็นโครงสร้างข้อมูลแบบใด (จงเขียน typedef ของ Monster)

```
typedef pair<string, pair<int,int>> Monster
```

b) (4 คะแนน) จงเขียนฟังก์ชัน vector<Monster> findAttackGE(vector<Monster> v, int atk) ฟังก์ชันนี้ รับเวคเตอร์ที่เก็บมอนสเตอร์และรับค่า atk แล้ว รีเทิร์นเวคเตอร์ที่เก็บมอนสเตอร์จาก v ที่ค่าพลังโจมตี มากกว่าหรือเท่ากับค่า atk

```
vector<Monster> findAttackGE(vector<Monster> v, int atk){
    vector<Monster> result;
    for(auto e:v){
        if (e.second.first >= atk)
            result.push_back(e);
    }
    return result;
}
```

c) (5 คะแนน) จงเขียนฟังก์ชัน void removeDefBetween(int startValue, int endValue, vector<Monster> &v) ทำการลบมอนสเตอร์ที่มีค่าพลังป้องกัน อยู่ระหว่าง startValue กับ endValue (รวมค่าของ startValue กับ endValue ด้วย) ออกจากเวคเตอร์ที่เราให้เป็นพารามิเตอร์

```
void removeDefBetween(int startValue, int endValue, vector<Monster> &v){
    vector<Monster> result;
    for(auto e:v){
        if (e.second.second < startValue || e.second.second > endValue)
            result.push_back(e);
    }
    v = result;
}
```

10. (10 คะแนน) จงเขียนฟังก์ชั่น stack<int> topNBottomM(stack<int> s1, stack<int> s2, int n, int m) ซึ่งคืนค่า stack ใหม่ ที่เกิดจากการเอาสมาชิกบนสุด n ตัว ของ s1 มาวางทับ สมาชิกล่างสุด m ตัว ของ s2 โดย n และ m ต้องมีค่าตั้งแต่ 0 ขึ้นไป เสมอ ถ้า n หรือ m มีค่ามากกว่าจำนวนสมาชิกใน stack ให้ทำเสมือนกับว่าค่านั้น ๆ มีค่าเท่ากับจำนวนสมาชิกใน stack

ตัวอย่างเช่น หากเรากำหนดให้ stack s1 มีค่าเป็น {1,2,3,4,5} และ s2 เป็น {16,17,18,19,20} โดยที่ถือว่าตัวซ้ายสุดในรายการเป็น top-of-stack การเรียก topNBottomM(s1,s2, 2, 4) จะได้ผลเป็น {1,2,17,18,19,20} หรือ การเรียก topNBottomM(s1, s2, 7, 7) จะได้ผลเป็น {1,2,3,4,5,16,17,18,19,20}

```
stack<int> topNBottomM(stack<int> s1, stack<int> s2, int n, int m) {
    stack<int> result;
    vector<int> temp;
    while(n-- || !s1.empty){
        temp.push_back(s1.top()); s1.pop();
    }
    while(s2.size > m)
        s2.pop();
    while(m-- || !s2.empty){
        temp.push_back(s2.top()); s2.pop();
    }
    for(int i=temp.size()-1; i>=0; i--){
        result.push(temp[i]);
    }
    return result;
}
```

11. (8 คะแนน) ในข้อนี้เป็นการเพิ่มความสามารถให้กับคลาส CP::queue โดยให้เพิ่มฟังก์ชันต่อไปนี้
    11.1. ฟังก์ชัน void printQueueFromBack() const ซึ่งจะพิมพ์ข้อมูลทั้งหมดของ queue จากท้ายคิวไปยังหัวคิวตามลำดับ โดยไม่ทำการเปลี่ยนแปลงค่าใน queue
    11.2. ฟังก์ชัน reverse ซึ่งมีผลทำให้มีการเรียงลำดับในคิวใหม่เสมือนกับว่าคิวมีการกลับทิศจากท้ายแถวกลายเป็นหัวแถว และ หัวแถวกลายเป็นท้ายแถว ตัวอย่างเช่น หากคิวมีข้อมูลเป็น {1,2,3,4,5,6} โดยให้ถือว่าตัวซ้ายสุดเป็นหัวแถว และตัวขวาสุดเป็นท้ายแถว การเรียก reverse จะทำให้คิวกลายเป็น {6,5,4,3,2,1}

```
template <typename T>
class vector {
  protected:
    T *mData; size_t mCap, mSize, mFront;   // ห้ามประกาศ data member เพิ่มเติม
  public:
  // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง โดยฟังก์ชันอื่น ๆ มิได้ระบุไว้เพื่อประหยัดหน้ากระดาษ ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
  // ให้เขียนเติมฟังก์ชันด้านล่างนี้ ให้ทำงานให้ถูกต้อง
    void printQueueFromBack() {
        int last =((mFront + mSize) % mCap)- 1;
        for(int i=0; i< mSize;i++){
           if(last<0) last+=mCap
           cout<< mData[last]<<' ';
           last--;
        }


    }

    void reverse() {
       T *arr = new T[mCap]();
       int last =((mFront + mSize) % mCap)- 1;
       for(int i=0; i< mSize;i++){
          if(last<0) last+=mCap
          arr[i]= mData[last];
          last--;
       }
       delete[] mData;
       mData = arr;
       mFront = 0

    }
};
```

12. (10 คะแนน) เนื่องจากสถานการณ์ COVID-19 เราต้องการสร้างระบบสำหรับการทำ Social Distancing ด้วยการเก็บข้อมูลว่า ณ ปัจจุบันนั้น อาคารต่าง ๆ มีใครอยู่ภายในบ้าง และให้ผู้ใช้สามารถ check-in หรือ check-out ออกจากอาคารต่าง ๆ ได้ และสามารถตอบได้ว่า ณ เวลาปัจจุบันมีใครอยู่ในอาคารนี้บ้าง โดยเราจะต้อง**ออกแบบและเขียนคลาส inside_building** ซึ่งทำหน้าที่ดังกล่าว

    เนื่องจากคลาสนี้จะมีการอ้างอิงถึงเวลาปัจจุบัน ขอให้ถือว่ามีฟังก์ชัน int get_time() ซึ่งจะคืนเวลาปัจจุบันมาให้ โดยมีหน่วยเป็นจำนวนวินาทีที่ผ่านมาตั้งแต่ 1 ม.ค. 2563 ให้สังเกตว่า การเรียก get_time แต่ละครั้งนั้น ค่าที่ได้จะไม่ลดลง

    ปัญหาของระบบนี้คือ คน “ส่วนใหญ่” ยอม check-in แต่แทบไม่มีใคร check-out เลย ดังนั้นเราจึงกำหนดว่า หากเวลา check-in นั้นนานเกินกว่า 2 ชม. (7200 วินาที) จากเวลาปัจจุบัน ให้ถือว่าคนคนนั้นไม่ได้อยู่ในตึกนั้นแล้ว (ถึงแม้ว่าคนคนนั้นจะไม่ได้ check-out ก็ตาม)

    คลาส inside_building ต้องมีฟังก์ชันต่อไปนี้เป็นอย่างน้อย (นิสิตสามารถเขียนฟังก์ชันอื่นเพิ่มเติมได้)

- check_in(string building, string name) ซึ่งถูกเรียกทันทีที่คนชื่อ name ทำการ scan QR Code เพื่อเข้าตึกชื่อ building
- check_out(string building, string name) ซึ่งถูกเรียกทันทีที่คนชื่อ name ทำการ scan QR Code เพื่อออกตึกชื่อ building
- int how_many(string building) นับจำนวนคนที่อยู่ในตึกดังกล่าว (ไม่รวมคนที่ check-out ไปแล้ว และ ไม่รวมคนที่ check-in เกินกว่า 2 ชม. ที่ผ่านมา)

โดยให้ถือว่าสมมติฐานเหล่านี้เป็นจริง

- คนแต่ละคนไม่มีใครที่ชื่อซ้ำกันเลย

- ในการเรียก check_in หรือ check_out แต่ละครั้งนั้นจะเป็นไปตามลำดับเวลาที่คนดังกล่าวทำการเช็คอินหรือเช็คเอ้าท์ และเป็นข้อมูลที่ถูกต้อง กล่าวคือ จะมีการเรียก check_in หรือ check_out ได้หากว่า คนคนนั้นมาที่ตึกนั่นหรือออกจากตึกนั้นจริง แต่มันก็เป็นไปได้ที่จะมีการเรียก check_in ของนาย A โดยที่นาย A ไม่ได้เรียก check_out หลังจากนั้น และเป็นไปได้ที่นาย A จะเรียก check_out โดยไม่เคยเรียก check_in มาก่อน (เนื่องจากคนคนลืมทำการ check_in หรือ check_out)
- เป็นไปได้ที่จะมีคนมากกว่า 1 คนทำการ check in หรือ check out ในเวลาเดียวกัน ณ ตึกเดียวกัน
- ในการใช้งานคลาสนี้ มีจำนวนครั้งในการเรียก check_in, check_out และ has_many เป็นจำนวนมาก และเป็นจำนวนใกล้เคียงกัน และ ให้ถือว่าคลาสนี้จะถูกใช้งานในระบบซอฟต์แวร์ที่ทำงานตลอดเวลาเป็นระยะเวลามากกว่า 1 ปี

จงตอบคำถามต่อไปนี้

12.1. จงระบุชื่อและประเภทของ Data Member ที่ใช้ในคลาสนี้ พร้อมทั้งระบุวัตถุประสงค์ของ Data Member ดังกล่าว

```
map<string, map<string,int>> stamp;  store timestamp of check-in time of each person in each building
map<string, set<string>>     people;  store set of person name inside a building
```

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

```
class inside_building{
  protected:
    map<string, map<string,int>> stamp;
    map<string, set<string>>     people;
  public:
    void check_in(string building, string name) {
        stamp[building][name] = get_time();
        people[building].insert(name);
    }
    void check_out(string building, string name){
        if(people[building].find(name)==people[building].end())
            return;
        stamp[building].erase(name);
        people[building].erase(name);
    }
    int int how_many(string building) {
        vector<string> forget;
        int now=get_time();
        for(auto e:people[building]){
            if(now-stamp[building][e]>7200)
                forget.push_back(e);
        }
        for(auto e:forget)
            check_out(building,e);
        return people[building].size();
    }
};
```

12.3. จงระบุข้อเสียของการออกแบบที่ได้ตอบในข้างต้น (ถ้าหากว่าคิดว่าคลาสที่ออกแบบมานั้นไม่มีข้อเสีย ให้เขียนว่า “ไม่มี”)

how_many ค่อนข้างช้า (O(n))

STL Reference

Common (All classes support these two capacity functions)

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

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

Iterator	iterator begin(); // an iterator referring to the first element
	iterator end(); // an iterator referring to the *past-the-end* element

vector<ValueT> และ list<ValueT>

Element Access สำหรับ vector	ValueT& operator[] (size_t n); ValueT& at(inti dx);

Modifier ที่ใช้ได้ทั้ง list และ vector	void push_back(const ValueT& val);
	void pop_back();
	iterator insert(iterator position, const ValueT& 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);

Modifier ที่ใช้ได้เฉพาะlist	void push_front(const ValueT& val);
	void pop_font;
	void remove(const ValueT& val);

set<ValueT>

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