FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2110211 Introduction to Data Structures

YEAR II, Second Semester, Midterm Examination, March 3, 2017, Time 13:00 – 16:00

ชื่อ-นามสกุล_____เลขประจำตัว | | | | | | | | | 2 | 1 | CR58_____

หมายเหตุ

1. ข้อสอบมีทั้งหมด 10 ข้อในกระดาษคำถามคำตอบจำนวน 8 แผ่น 8 หน้า คะแนนเต็ม 79 คะแนน
2. ไม่อนุญาตให้นำตำราและเครื่องคำนวณต่างๆ ใดๆ เข้าห้องสอบ
3. ควรเขียนตอบด้วยลายมือที่อ่านง่ายและชัดเจน สามารถใช้ดินสอเขียนคำตอบได้
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่ผู้คุมสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบออกจากห้องสอบ ข้อสอบเป็นทรัพย์สินของราชการซึ่งผู้ลักพาอาจมีโทษทางคดีอาญา
6. ผู้ที่ประสงค์จะออกจากห้องสอบก่อนหมดเวลาสอบ แต่ต้องไม่น้อยกว่า 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. ผู้ที่ปฏิบัติเข้าข่ายทุจริตในการสอบ ตามประกาศคณะวิศวกรรมศาสตร์
   มีโทษ คือ ได้รับ สัญลักษณ์ F ในรายวิชาที่ทุจริต และพักการศึกษาอย่างน้อย 1 ภาคการศึกษา

รับทราบ

ลงชื่อนิสิต (.....)

---

1. (4 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 ต้องการที่เก็บข้อมูลเพื่ออำนวยความสะดวกในการเขียนเมท็อด **`string getCarOwnerName(string plate)`** เพื่อขอชื่อเจ้าของรถ โดยหาจากป้ายทะเบียน

.....

1.2 ต้องการที่เก็บข้อมูล สำรับไพ่ ในขณะที่กำลังเล่นไพ่ (ไพ่จะโดนจั่วจากสำรับทีละใบ) ให้ข้อมูลของไพ่ ประกอบด้วย ดอกไพ่ และ ค่า 2 ถึง J Q K A

.....

1.3 ต้องการที่เก็บข้อมูล คำ และความถี่ของคำนั้น ๆ จากหนังสือ โดยที่ข้อมูลเรียงจากความถี่น้อย ไปมาก

.....

1.4 ต้องการเก็บข้อมูลแบบบ้าน ให้สามารถค้นหาได้ด้วยจำนวนห้องนอน (เจอทีเดียวหลายแบบบ้านได้) แบบบ้านแต่ละแบบประกอบด้วยชื่อแบบ พื้นที่ใช้สอย และจำนวนรถที่จอดได้

.....

2. (5 คะแนน) จงบรรยายว่า โค้ดต่อไปนี้ กำลังทำอะไร ในพื้นที่ว่างใต้โค้ดที่ให้มา

2.1
```
void f1(vector<int> &v) {
    vector<int> v2;
    int vectorSize = v.size();
    for(vector<int>::reverse_iterator i = v.rbegin(); i != v.rend(); i++){
        v2.push_back(*i);
    }
    v = v2;
}
```

.....

2.2
```
vector<pair<int,string>> f2(vector<pair<int,string>> v, int x) {
    int vectorSize = v.size();
    vector<pair<int,string>> answer;
    for(vector<pair<int,string>>::iterator i = v.begin(); i != v.end(); i++){
        if((*i).first == x){
            answer.push_back(*i);
        }
    }
    return answer;
}
```

.....

2.3
```
set<int> f3(set<int> s1, set<int> s2) {
     set<int> s3;
     for(set<int>::iterator i = s1.begin(); i != s1.end(); i++){
        set<int>::iterator p = s2.find(*i);
        if(p == s2.end()){
            s3.insert(*i);
        }
     }
     return s3;
}
```

.....

2.4
```
void f4(map<string,string> m, string word){
    for (map<string,string>::iterator it = m.begin(); it != m.end(); it++) {
        if((*it).second > word){
            cout << "(" << (*it).first << " , " << (*it).second << ")";
        }
    }
    cout << endl;
}
```

.....

2.5
```
void f5(deque<int> &a, int i, int v) {
    int s = a.size();
    if(i >s || i<0 )  return;
    for(int j=0; j<i && j<s; j++){
        a.push_back(a.front());
        a.pop_front();
    }
    a.push_front(v);
     for(int j=0; j<i; j++){
        a.push_front(a.back());
        a.pop_back();
    }
}
```

.....

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

********** สำคัญ!!! **********

ตั้งแต่ข้อ 4 เป็นต้นไปเป็นการเขียนโปรแกรม การเขียนโปรแกรมจะต้องไม่ผิด syntax ในส่วนที่สำคัญ โดยเฉพาะอย่างยิ่งการเรียก function และ argument ของฟังก์ชันจะต้องถูกต้อง นอกจากนี้คะแนนจะแปรผันตาม "ประสิทธิภาพ" ของโปรแกรมที่เขียนมา

****** ให้สังเกตถึงข้อกำหนดต่าง ๆ ในโจทย์ให้รอบคอบ <u>การไม่ทำตามข้อกำหนดจะมีผลต่อคะแนนเป็นอย่างมาก</u> ******

3. (5 คะแนน) ในข้อนี้ให้นิสิตเขียนฟังก์ชั่น removeThirdMax(priority_queue<int>& pq) ที่เอาตัวที่มากที่มากที่สุดเป็นลำดับที่สาม ออกจาก priority queue โดยหลังจบฟังก์ชั่นข้อมูลตัวอื่นๆยังต้องคงอยู่เหมือนเดิม ตัวอย่างเช่น หากตอนเริ่ม priority queue มี 1 3 5 7 8 อยู่ หลังเรียกฟังก์ชั่นนี้จะมีข้อมูลเป็น 1 3 7 8 อยู่ (เนื่องจาก 5 เป็น ตัวที่มากที่มากที่สุดเป็นลำดับที่สาม) โดยรับประกันว่า priority queue มีข้อมูลตั้งแต่ 3 ตัวขึ้นไปเสมอ

```
void removeSecondMax(priority_queue<int>& pq) {
       // เติม C++ code ตรงนี้









}
```

4. (5 คะแนน) ในข้อนี้ให้ ให้เขียน comparator ชอง class Student ที่เมื่อใช้กับ sort แล้วจะ เรียง ตาม อายุ(age) จากมากไปน้อย ถ้าอายุเท่าเรียงตาม ชื่อ(fname) จากน้อยไปมาก ถ้าเท่าก็ตามนามสกุล(lname) จากน้อยไปมาก

```
class Student{
public:
    int age;
    string fname, lname;
};
class StudentComparator{
    bool operator() (const Student& s1, const Student& s2) const {
      // เติม C++ code ตรงนี้
















    }
};
int main() {
      vector<Student> students;
      //……
      sort(students.begin(), students.end(), StudentComparator());
}
```

5. (5 คะแนน) ในข้อนี้นิสิตได้รับ ฟังก์ชั่นทางคณิตศาสตร์บนตัวเลขจำนวนเต็มไปยังจำนวนเต็มในรูป f(x), g(f), h(g) มาในรูปแบบของ map ให้นิสิตสร้างและ return map ที่แทนฟังก์ชั่น A(x) ซึ่งนิยามโดย A(x) = h(g(f(x))) โดยเรารับประกันว่าสามารถหา h(x) จากข้อมูลที่มีได้เสมอ

```
map<int,int> get_h(const map<int, int>& f_x, const map<int, int>& g_f, const map<int, int>& h_g) {
          map<int,int> h_x;
```

```
   // เติม C++ code ตรงนี้




   return h_x;
}
```

6. (10 คะแนน) ในข้อนี้นิสิตมี vector<int> v1, v2 เก็บเลขเรียงจากมากไปน้อยไม่ซ้ำภายใน v1, v2 เอง (แต่อาจจะมีตัวเลขที่อยู่ในทั้ง v1 และ v2 ได้) ให้นิสิตเขียน ฟังก์ชั่นที่คืนค่า intersection ของ v1 และ v2 (ข้อมูลที่อยู่ในทั้ง v1 และ v2)

```
vector<int> get_intersect(const vector<int>& v1, const vector<int>& v2) {
     vector<int> result;
     // เติม C++ code ตรงนี้
     return result;
}
```

7. (10 คะแนน) ในข้อนี้ให้นิสิตเพิ่มบริการ ของ CP::vector insert_to(iterator it, const vector<T>& v) โดยให้เอาข้อมูลใน v ไปแทรกหน้า it ของ this, โดยห้ามเรียก insert ของ CP::vector (แต่เรียกคำสั่งอื่นๆของ CP::vector ได้) และห้ามใช้ stl ใดๆ

```
template <typename T>
class vector {
  protected:
    T      *mData;     size_t  mCap;      size_t  mSize;
    void rangeCheck(int n)
    void expand(size_t capacity)
    void ensureCapacity(size_t capacity)
  public:
    { ... } // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง โดยฟังก์ชันอื่น ๆ มิได้ระบุไว้เพื่อประหยัดหน้ากระดาษ ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
    void insert_to(iterator it, const vector<T>& v) {
      // เติม C++ code ตรงนี้
    }
}
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

8. (10 คะแนน) มีร้านอาหารร้านหนึ่งซึ่ง มีคนต้องการไปใช้บริการเป็นจำนวนมาก จึงเปิดให้มีการลงทะเบียนล่วงหน้าผ่านทาง website โดยให้กรอกหมายเลขโทรศัพท์ อย่างไรก็ตาม มีบางคนจองคิวซ้ำ ๆ ขึ้นมา โดยกรอกหมายเลขโทรศัพท์เดิมเข้าไปหลายครั้ง (อาจจะไม่ได้กดต่อเนื่องกันก็เป็นได้ เช่น กดตอนเช้า แล้วก็กดตอนบ่ายอีกทีหนึ่ง) กำหนดให้มี queue<int> q อยู่ ซึ่ง q นั้นเก็บหมายเลขโทรศัพท์ตามลำดับของคนที่มาจอง (ซึ่งอาจจะมีข้อมูลซ้ำได้) จงเขียนฟังก์ชันเพื่อทำให้ข้อมูลใน queue นั้นไม่มีตัวที่ซ้ำอยู่เลย โดยถ้ามีใครที่จองซ้ำ ให้ลบการจองครั้งหลัง ๆ ออกไป (ตัวอย่างเช่นในคิวมี <4,1,2,3,1,2,1,5> เมื่อเรียกฟังก์ชันแล้ว ต้องกลายเป็น <4,1,2,3,5>

```
void make_unique(queue<int> &q) {
  // เติม C++ code ตรงนี้









}
```

9. (10 คะแนน) ในโครงสร้างข้อมูล CP::Stack นั้นมี data member อยู่ 3 ตัวคือ mSize, mData, mCap ในโจทย์ข้อนี้ ต้องการให้นิสิตเขียนโครงสร้างข้อมูล CP::Stack ขึ้นใหม่ โดยมีข้อกำหนดคือ ในโครงสร้างข้อมูลนั้น จะมี data member ได้เพียงชนิดเดียว คือ set<pair<int,T>> จากโค้ดด้านล่างนี้ ให้นิสิตเติมฟังก์ชัน push, pop, top ให้ทำงานตามที่ stack ควรจะเป็น ภายใต้เงื่อนไขดังกล่าว **ห้ามนิสิตเพิ่ม data member อื่นใดนอกเหนือจาก set ที่ได้กำหนดไว้แล้ว**

```
template <typename T>
class stack {
  protected:
    set<pair<int,T>> mData;  // ห้ามประกาศ data member เพิ่มเติม
  public:
    bool empty() const { return mData.empty(); }
    size_t size() const { return mData.size();  }

    // ให้เขียนเติมฟังก์ชันด้านล่างนี้ ให้ทำงานให้ถูกต้อง
    const T& top() const{









    }

    void push(const T& element) {









    }

    void pop() {









    }
};
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

10. (15 คะแนน) โจทย์ข้อนี้เป็นการเขียนคลาสโครงสร้างข้อมูลที่จำลองการทำงานของระบบดูแลห้องสมุดซึ่งถูกใช้โดยเจ้าหน้าที่ของห้องสมุด ในระบบห้องสมุดนั้นมีหนังสืออยู่หลายรายการ รายการละหลายชุด (หลาย copy) หนังสือแต่ละรายการสามารถระบุได้ด้วย ชื่อเรื่อง (title) และ ชื่อผู้แต่ง (author) ชื่อเรื่องและชื่อผู้แต่งนั้นประกอบด้วยตัวหนังสือตัวภาษาอังกฤษและช่องว่างเท่านั้น หนังสือแต่ละรายการจะมี “หมายเลข” เป็นบาร์โค้ดเขียนไว้ที่หน้าปก บาร์โค้ดนี้ออกโดยระบบห้องสมุด หนังสือรายการเดียวกัน ไม่ว่าจะมีกี่ชุดก็ตาม จะมีบาร์โค้ดเป็นเลขเดียวกัน

ระบบห้องสมุดมีงานหลักต้องทำอยู่ 4 อย่าง คือ การสอบถามจำนวนชุด, การสืบค้นหนังสือโดยผู้แต่ง, การลงทะเบียนหนังสือเล่มใหม่, และ การยืม/คืน โดยมีรายละเอียดดังนี้ 1) การสอบถามจำนวนชุด คือการตรวจสอบว่าหนังสือรายการที่ต้องการนั้น มีพร้อมให้ยืมอยู่ ณ ห้องสมุดอยู่กี่ชุด (ไม่นับชุดที่ถูกยืมออกไปแล้ว) 2) การสืบค้นหนังสือโดยผู้แต่ง คือบริการที่ต้องการทราบว่า ณ ปัจจุบัน หนังสือทุกเรื่องของผู้แต่งดังกล่าว มีอยู่ในห้องสมุดพร้อมให้ยืมอยู่กี่ชุด 3) การลงทะเบียนหนังสือเล่มใหม่ เป็นการนำเอาหนังสือเล่มใหม่ (ซึ่งห้องสมุดอาจจะเคยมีรายการดังกล่าวอยู่แล้วหรือไม่ก็ได้) เข้ามาไว้ในห้องสมุดเพื่อให้ยืม 4) การยืม/คืน ซึ่งกระทำโดยผู้ใช้บริการจะหยิบหนังสือมายื่นให้เจ้าหน้าที่ห้องสมุด และเจ้าหน้าที่ห้องสมุดจะทำการบันทึกการยืม/คืนดังกล่าวเพื่อปรับปรุงจำนวนหนังสือรายการดังกล่าวที่มีอยู่ในห้องสมุด เพื่อความรวดเร็วในการยืม/คืน เจ้าหน้าที่จะใช้บาร์โค้ดในการบันทึกข้อมูล

จงเขียนคลาส BookLibrary ซึ่งมีบริการต่อไปนี้

- int current_stock(string author, string title) เป็นการสอบถามซึ่งจะต้องคืนจำนวนชุดของหนังสือที่มีชื่อผู้แต่ง และ ชื่อเรื่องตามที่ระบุที่มีอยู่ในห้องสมุดพร้อมให้ยืม ณ ปัจจุบัน ในกรณีที่ห้องสมุด “ไม่เคยมี” หนังสือเล่มดังกล่าวอยู่เลย ต้องคืนค่า -1
- vector<pair<string,int>> books_by_author(string author) เป็นการสอบถามว่า หนังสือทั้งหมดของผู้แต่งที่ระบุนั้น แต่ละชื่อเรื่องมีอยู่ในห้องสมุดพร้อมให้ยืมอยู่กี่เล่ม โดยให้คืนเป็น vector ของ pair ของ ชื่อเรื่อง และ จำนวนพร้อมให้ยืม ถ้าไม่มีหนังสือของผู้แต่งดังกล่าว ให้คืน vector ว่าง
- void registerNewBook(string author, string title) เป็นการลงทะเบียนหนังสือเล่มใหม่ ซึ่งมีชื่อผู้แต่ง ชื่อเรื่องตามที่กำหนด หนังสือดังกล่าวอาจจะไม่เคยปรากฏมาก่อนในห้องสมุดก็เป็นได้ เมื่อลงทะเบียนแล้ว หนังสือเล่มดังกล่าวจะพร้อมให้ยืมทันที ณ ตอนเริ่มต้นที่ใช้ระบบดูแลห้องสมุดนั้น ห้องสมุดไม่มีหนังสือใดอยู่เลย เมื่อทำการลงทะเบียนหนังสือเล่มใด ๆ ก็ตาม ถ้าหากหนังสือรายการดังกล่าวไม่เคยมีอยู่เลย ก็จะมีการกำหนดบาร์โค้ดให้กับหนังสือเล่มดังกล่าว โดยบาร์โค้ดจะเรียงลำดับไปเรื่อย ๆ ตั้งแต่เลข 0, 1, 2, 3 ไปเรื่อย ๆ ตามลำดับที่ลงทะเบียน ถ้าหากหนังสือรายการดังกล่าวเคยมีอยู่แล้ว ก็จะไม่มีการกำหนดบาร์โค้ดเพิ่มเติม
- void book_borrow(int barcode) เป็นการลงทะเบียนการยืมหนังสือตามบาร์โค้ดดังกล่าว
- void book_return(int barcode) เป็นการลงทะเบียนการคืนหนังสือตามบาร์โค้ดดังกล่าว

ในโจทย์ข้อนี้ การทำงานนั้นจะต้องถูกต้อง และ รวดเร็ว แต่ถ้าหากทำให้รวดเร็วไม่ได้ ก็ขอให้พยายามทำให้ถูกต้องตามข้อกำหนดมากที่สุด ให้ออกแบบคลาสโดยคำนึงว่าจำนวนหนังสือนั้นอาจจะมีมากมายเต็มไปหมด และมีการยืม/คืนบ่อยครั้งมาก ๆ

10.1 จงอธิบาย data member ต่าง ๆ ที่ใช้ในคลาสที่ออกแบบขึ้น ว่ามี member อะไรบ้าง และ แต่ละตัวทำหน้าที่อะไร ทำไมถึงเลือกใช้ประเภทตัวแปรดังกล่าว

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

10.2 จงเขียนคลาส Library ตามข้อกำหนดที่ได้ระบุไว้ข้างต้น ถ้าหากเนื้อที่ไม่พอเขียน ให้เขียนไว้ด้านหลังของ หน้า 7 เท่านั้น

STL Reference

(be noted that the underlined method is not available in CP version)

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

Class vector<ValueT>

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

Modifier	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);

Class 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);

Class 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

Class queue<ValueT>

Element Access	ValueT front(); ValueT back();

Class stack<ValueT>

Element Access	ValueT top();

Class priority_queue<ValueT, ContainerT = vector<ValueT>, CompareT = less<ValueT> >

Element Access	ValueT top();

Useful functions

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