FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2110211 Introduction to Data Structures

YEAR II, Second Semester, Midterm Examination, December 24, 2013, Time 13:00 – 16:00

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 11 ข้อในกระดาษคำถามคำตอบจำนวน 9 แผ่น 9 หน้า คะแนนเต็ม 90 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. (9 คะแนน) จงตั้งชื่อฟังก์ชัน (ที่ได้ความหมายเดียวกับที่ทำงาน) พร้อมทั้งเขียนอธิบายเพิ่มเติมอีกเล็กน้อยได้ในช่องว่างด้านขวา

```
bool _____(vector<int>& v) {
    set<int> s;
    s.insert(v.begin(), v.end());
    return s.size() == v.size();
}
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

```
void _____(int d[], int n) {
    map<int,int> m;
    for (int i=0; i<n; i++) m[d[i]]++;
    map<int,int>::iterator it = m.begin();
    for (int i=0; i<n; ++it) {
        for (int j=0; j<(*it).second; j++,i++) {
            d[i] = (*it).first;
        }
    }
}
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

```
void _____( vector<int>& vi, int k,
                          vector<int>& vo ) {
    priority_queue<int> pq;
    for (int i=0; i<k&&i<vi.size(); i++)
      pq.push(vi[i]);
    for (int i=k; i<vi.size(); i++) {
        if (vi[i]<pq.top()) {
            pq.pop();
            pq.push(vi[i]);
        }
    }
    vo.clear();
    while(!pq.empty()) {
        vo.push_back(pq.top());
        pq.pop();
    }
}
```

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

2. (8 คะแนน) จงเขียนรายละเอียดการทำงานของฟังก์ชันต่อไปนี้ให้ตรงตามที่เขียนใน comment

```
void removeAllS2FromS1( set<int>& s1, set<int>& s2 ) {
// ลบสมาชิกทุกตัวของ s2 ที่ปรากฏใน s1 ออกจาก s1 เช่น s1={1,2,3,4}, s2={2,3,9} ลบแล้วได้ s1 = {1,4}




}
```

```
int bottom(stack<int>& s) {
// ลบ และ คืนข้อมูลที่อยู่ที่ตำแหน่งข้างล่างสุดของสแตก




}
```

3. (4 คะแนน) มีคลาส **Donor** เก็บรายละเอียดผู้บริจาคเงินกองทุนตึก 100 ปี วิศวฯ ได้เขียน **operator<** ใน **Donor** เพื่อให้ผู้บริจาค **a** มี "ค่าน้อยกว่า" **b** เมื่อ **a** บริจาคจำนวนเงินน้อยกว่า **b** เนื่องจากชอบมีผู้สอบถามอยู่บ่อยๆ ว่า ตอนนี้ใครเป็นผู้บริจาคเงินมากสุด *m* อันดับแรก จึงตัดสินใจมี *priority_queue* เก็บข้อมูลผู้บริจาคทั้งหลาย จงเขียนฟังก์ชัน **getKMaxDonors** ที่รับ *priority_queue* ที่ว่านี้ และคืน *vector* ของผู้บริจาคเงินมากสุด *m* อันดับแรก (เรียงจากมากไปน้อยจากซ้ายไปขวาใน *vector*)

```
vector<Donor> getKMaxDoners(priority_queue<Donor> & donors, int m) {




}
```

4. (3 คะแนน) เมื่อสั่งใน **main** ของโปรแกรมทางขวานี้ทำงานแล้ว จะแสดงเครื่องหมาย * (จากคำสั่ง **cout << "*";**) กี่ตัว อธิบายที่มาของคำตอบในช่องว่างทางขวาด้วย

```
int f(int n, map<int,int>& F) {
    if (n < 2) return n;
    if (F[n] != 0) return F[n];
    cout << "*";
    return F[n] = f(n-1, F) + f(n-2, F);
}
int f(int n) {
    map<int,int> F;
    return f(n, F);
}
int main() {
    cout << f(5) << endl;
    return 0;
}
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

5. (6 คะแนน) ตอบคำถามต่อไปนี้สั้น ๆ ว่า แต่ละปัญหาต้องมีที่เก็บข้อมูลประเภทใด

   0. ต้องการเก็บรายชื่อนิสิตคณะวิศวฯทุก ๆ รุ่น แต่ละรุ่นมีหมายเลขรุ่นกำกับ เพื่อเขียนฟังก์ชัน
      `int getClassID(string name)` ที่คืนหมายเลขรุ่นของคนชื่อ **`name`**
      ตอบ: `map<string, integer>` *key* คือชื่อ *mapped value* คือหมายเลขรุ่น (ข้อนี้เป็นตัวอย่าง)

   1. ต้องการเขียนฟังก์ชัน **`bool hasCD(CD & x)`** เพื่อตรวจว่า เรามีแผ่นซีดี **`x`** เก็บอยู่หรือไม่

      .....

   2. ในการสมัครสอบเข้าศึกษาตรงในจุฬาฯ ผู้สมัครต้องเลือกสาขาที่ต้องการ เลือกได้ 4 ลำดับมากสุด หลังการกรอกข้อมูลพร้อมกับรหัสสาขาที่เลือก ระบบจะกำหนดหมายเลขการสมัคร (**`appID`**) ให้กับผู้สมัครแต่ละคน เราควรมีที่เก็บข้อมูลอะไร เพื่อเขียนฟังก์ชัน **`string getAppProg(string appID, int k)`** เพื่อคืนรหัสสาขาในลำดับที่ **`k`** ที่ผู้สมัคร **`appID`** ได้เลือกไว้

      .....

   3. มีข้อมูลคะแนนสอบของวิชา 2110101 ย้อนหลัง 30 ปี (เช่น 5331010121 ได้ 98.2, 4730002021 ได้ 50.333, …) ต้องการเขียนฟังก์ชันให้บริการสอบถามว่า มีนักเรียนกี่คนได้คะแนนสอบเท่ากับ x (ไม่สนใจว่าใครได้คะแนนเท่าไร)

      .....

6. (5 คะแนน) ในข้อนี้ ให้นิสิตเพิ่มบริการใหม่ที่ชื่อว่า void getReverse(const stack<T>& s) ให้กับคลาส CP::stack โดยบริการนี้จะทำการลบข้อมูลของกองซ้อนปัจจุบันออกทั้งหมดแล้วนำข้อมูลจากกองซ้อน s ใส่เข้ามาแทนในลำดับกลับหลัง เช่น หาก s มี 5 1 8 9 อยู่ ข้อมูลที่เก็บอยู่ในกองซ้อนปัจจุบันหลังจากการเรียก getReverse จะเป็น 9 8 1 5

ห้ามนิสิตใช้ stl หรือคลาสอื่นใดในข้อนี้ แต่สามารถเรียกฟังก์ชันอื่น ๆ ของ CP::stack ได้ตามปรกติ

```
template <typename T>
class stack {
protected:
   T      *mData;
   size_t  mCap;
   size_t  mSize;
    void expand(size_t capacity) { ... }
    void ensureCapacity(size_t capacity) { ... }
public:
    stack(const stack<T>& a) { ... }
    stack() { ... }
    ~stack() { ... }
    bool empty() const { ... }
    size_t size() const {...}
    const T& top() { ... }
    void push(const T& element) { ... }
    void pop() { ... }

    void getReverse(const stack<T>& s) {
         // เติมคำตอบที่นี่ นิสิตสามารถใช้ตัวแปรสมาชิก หรือเรียก บริการต่างๆ ของ stack ตามที่มีข้างบนได้









    }
};
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

7. (10 คะแนน) ในข้อนี้ ให้นิสิตเพิ่ม constructor ใหม่ของ CP::queue โดย constructor นี้จะรับ vector ของแถวคอยเป็นข้อมูลนำเข้า แล้วนำข้อมูลของแถวคอยเหล่านั้นตามลำดับ First-in-first-out (FIFO) มาผลัดกันใส่ในแถวคอยปัจจุบันนี้จนกว่าจะหมดตัวอย่างเช่น

vector<queue<int> > qs(3);

qs[0].push(1); qs[0].push(3); qs[0].push(4); qs[0].push(8);

qs[1].push(2); qs[1].push(9);

qs[2].push(5); qs[2].push(7); qs[2].push(6);

queue<int> q(qs);

แถวคอย q จะมีข้อมูลเก็บอยู่ตามลำดับจากก่อนไปหลังดังนี้ 1 2 5 3 9 7 4 6 8

เรารับประกันว่า qs จะมีอย่างน้อยหนึ่งแถวคอย และทุกๆแถวคอย qs[i] จะมีข้อมูลอย่างน้อยหนึ่งตัว

ห้ามนิสิตใช้ stl หรือคลาสอื่นใดในข้อนี้ แต่สามารถใช้ฟังก์ชันต่าง ๆ ของ CP::vector และ CP:queue ได้ตามปรกติ

```
template <typename T>
class queue {
protected:
   T      *mData;
   size_t  mCap;
   size_t  mSize;
   size_t  mFront;
    void expand(size_t capacity) { ... }
    void ensureCapacity(size_t capacity) { ... }

public:
    queue(const queue<T>& a) { ... }
    queue() { ... }
    ~queue() { ... }
    bool empty() const { ... }
    size_t size() const {...}
    const T& front() { ... }
    const T& back() { ... }
    void push(const T& element) { ... }
    void pop() { ... }

    queue(const vector<queue<T> >& qs) {
          // เติมคำตอบที่นี่ นิสิตสามารถใช้ตัวแปรสมาชิก หรือเรียก บริการต่างๆ ของ queue ตามที่มีข้างบนได้ หรือของ vector ได้
    }
};
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

8. (5 คะแนน) จากคลาส CP::vector<T> ที่นิสิตได้เรียนรายละเอียดของโครงสร้างในวิชานี้ จงเขียนฟังก์ชัน vector<T> intersect(vector<T> a) ที่รับเวกเตอร์ a เพื่อประมวลผลให้ได้เวกเตอร์ใหม่ที่เป็นผลลัพธ์จากการทำอินเตอร์เซกชันของ this กับ a โดยจะไม่มีข้อมูลที่ซ้ำกันในผลลัพธ์นี้ ตัวอย่างเช่น ถ้า v = {1,1,3} และ v2 = {9,1,1} ผลลัพธ์ของการเรียก v1.intersect(v2) จะได้เป็น {1}

ห้ามใช้ฟังก์ชันของ algorithm ช่วยในการเขียน และอนุญาตให้เรียกใช้ฟังก์ชันของ CP::vector<T> ต่อไปนี้ได้เท่านั้น

- bool contains(T& element) ใช้ตรวจสอบว่ามี element อยู่ในเวคเตอร์หรือเปล่า
- void push_back(const T& element) ใช้ใส่ element เข้าไปเป็นข้อมูลสุดท้ายในเวคเตอร์

```
template <typename T>
class vector {
protected:
   T      *mData;
   size_t  mCap;
   size_t  mSize;

public:
   // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง ให้นิสิตเขียนบริการ intersect เพิ่มเติม โดยห้ามเรียกใช้ฟังก์ชันอื่นใดนอกจาก contains และ push_back
   // นิสิตสามารถเขียนฟังก์ชันเพิ่มเติมได้ ถ้าต้องการ
```

9. (10 คะแนน) ให้ปรับปรุงโครงสร้างข้อมูลประเภท CP::queue ดังที่เห็นในข้อ 7 โดยทำให้ queue นั้นกลายเป็นแถวคอยสองด้าน (หรือที่เรียกว่า Deque) โดยให้นิสิตเขียนบริการเพิ่มเติม บริการที่ต้องทำการแก้ไขเพิ่มเติมมีดังต่อไปนี้

    void push_front(const T& element) ใช้เพื่อใส่ element เข้าไปด้านหน้าของแถวคอยสองด้าน

    void push_back(const T& element) ใช้เพื่อใส่ element เข้าไปด้านหลังของแถวคอยสองด้าน

    void pop_front() ใช้เพื่อลบข้อมูลที่อยู่ด้านหน้าของแถวคอยสองด้านออก

    void pop_back() ใช้เพื่อลบข้อมูลที่อยู่ด้านหลังของแถวคอยสองด้านออก

    void print() ใช้เพื่อแสดงข้อมูลที่อยู่ในแถวคอยสองด้านโดยแสดงตามลำดับจากด้านหน้าไปหลังไปยังหน้าจอ

โดยมีข้อแม้ว่าบริการเดิมต่างๆของ queue ทุกบริการจะต้องยังทำงานถูกต้องอยู่ นิสิตสามารถเรียกใช้บริการใดๆที่มีอยู่ใน queue ได้โดยไม่ต้องเขียนใหม่ (ไม่จำเป็นต้องตรวจสอบกรณีที่มีการเอาข้อมูลออกเมื่อแถวคอยสองด้านนี้ไม่มีข้อมูล)

ห้ามนิสิตใช้ stl หรือคลาสอื่นใดในข้อนี้ แต่สามารถเรียกฟังก์ชันต่าง ๆ ของ CP::queue ได้ตามปรกติ

```
template <typename T>
class queue {
// ในคลาสนี้มีบริการอื่น ๆ ตามที่เขียนไว้ในข้อ 7 ทุกประการ
public:
    void push_front(const T& element) {









    }
    void pop_front() {









    }
    void push_back(const T& element) {









    }
    void pop_back() {









    }
    void print() {









    }
};
```

10. (10 + 5 คะแนน) เราอยากจะสร้างคลาส Matrix ซึ่งทำหน้าทีเก็บข้อมูลเมทริกซ์ในการคำนวณทางคณิตศาสตร์ กำหนดให้ $A_{r,c}$ หมายถึงค่าของสมาชิกของเมทริกซ์ดังกล่าวในแถวที่ r คอลัมน์ที่ c (r มีค่าตั้งแต่ 1 ถึงจำนวนแถว และ c มีค่าตั้งแต่ 1 ถึงจำนวนคอลัมน์ของเมทริกซ์) ให้นิสิตเขียน class ซึ่งมีบริการดังต่อไปนี้

- `Matrix(size_t row, size_t col)` เป็น constructor สำหรับเมทริกซ์ขนาด row แถว col คอลัมน์ และกำหนดค่าให้ทุก ๆ ช่องใน matrix มีค่าเป็น 0
- `void set(size_t r, size_t c, double x)` สำหรับการกำหนดค่าสมาชิกในแถว r คอลัมน์ c ให้มีค่าเป็น x โดย (โดย 1 <= r <= row และ 1 <= c <= col เมื่อ row และ col เป็นจำนวนแถวและจำนวนคอลัมน์ของเมทริกซ์ดังที่กำหนดไว้ใน constructor)
- `double get(size_t r, size_t c)` สำหรับการอ่านค่าสมาชิกในแถว r คอลัมน์ c (โดย 1 <= r <= row และ 1 <= c <= col)

- `void swap(size_t r1, size_t r2)` เป็นการสลับค่าของแถวที่ r1 กับแถวที่ r2 ในเมทริกซ์ (โดย 1 <= r1,r2 <= row)
- `void addMultiple(size_t r1, size_t r2, double val)` เป็นการปรับปรุงค่า $A_{r1,i}$ ให้เป็น $A_{r1,i} + val * A_{r2,i}$ สำหรับทุก ๆ ค่า i ตั้งแต่ 1 ถึง col (โดย 1 <= r <= row)

ในข้อนี้ ถ้านิสิตสามารถทำให้คำสั่ง **swap**, get, set นั้นใช้เวลาคงที่เสมอโดยไม่ขึ้นอยู่กับจำนวนคอลัมน์ พร้อมทั้งทำให้คำสั่ง addMultiple นั้นใช้เวลาแปรผันตรงกับจำนวนคอลัมน์ได้ จะได้คะแนนพิเศษเพิ่มอีก 5 คะแนน (เพิ่มขึ้นมาจากคะแนนเต็ม)

```
class Matrix {
protected: // ประกาศตัวแปรที่ต้องใช้ตรงนี้




public:
  Matrix(size_t row, size_t col) {









  }
  void set(size_t r, size_t c, double x) {



  }
  double get(size_t r, size_t c) {



  }
  void swap(size_t r1, size_t r2) {




  }
  void addMultiple(size_t r1, size_t r2, double val) {




  }
};
```

11. (15 คะแนน) เว็ปไซต์ขายสินค้าแห่งหนึ่งมีสินค้าอยู่ N แบบ สินค้าแต่ละแบบนั้นถูกกำกับด้วยรหัสสินค้า ซึ่งเป็นข้อมูลประเภท string ความยาว 10 ตัวอักษร สินค้าแต่ละแบบนั้นมีรหัสสินค้าที่แตกต่างกัน และมีราคาเป็นข้อมูลประเภท double ที่แตกต่างกัน เว็บไซต์ดังกล่าวมีบริการค้นหาสินค้าตามราคา โดยผู้ใช้สามารถกำหนดช่วงราคาของสินค้าที่ต้องการข้อมูล แล้วเว็บไซต์จะแสดงข้อมูลที่มีราคาที่ตรงกับช่วงราคาที่กำหนดมาให้ ตั้งแต่เปิดเว็บไซต์ขึ้นมา เจ้าของเว็บไซต์ได้ทำการบันทึกข้อมูลไว้สามอย่าง คือ
    1) รหัสสินค้าและราคาสินค้าแต่ละชิ้น อยู่ในตัวแปร vector<pair<string, double> > price โดย price[i].first คือรหัสสินค้าของสินค้าชิ้นที่ i ซึ่งมีราคาเป็น price[i].second
    2) ช่วงราคาที่ผู้ใช้ได้ทำการค้นหาสินค้าทั้งหมด อยู่ในตัวแปร vector<pair<double,double> > query โดย query[i] เก็บข้อมูลการค้นหาสินค้าหนึ่งครั้ง ซึ่งค้นหาในช่วงราคาตั้งแต่ query[i].first ถึง query[i].second

3) จำนวนสินค้าแต่ละแบบที่ขายไป อยู่ในตัวแปร vector< pair<string, int> > sale โดย sale[i] เก็บว่า สินค้าที่มีรหัสสินค้าเป็น sale[i].first นั้นขายได้ sale[i].second ชิ้น

อยู่มาวันหนึ่ง เจ้าของเว็บไซต์นี้ต้องการจะทำโปรโมชันส่งเสริมการขายสินค้าบางชิ้น โดยจะนำสินค้า N อันดับแรกที่ถูกค้นหาโดยผู้ใช้เป็นจำนวนครั้งมากที่สุด (ถ้าถูกค้นเป็นจำนวนครั้งเท่ากัน ให้เลือกสินค้าที่มีรหัสสินค้ามาก่อนตามพจนานุกรม) รวมกับ สินค้าที่ขายได้มากชิ้นที่สุด S ลำดับแรก (ถ้าขายได้เป็นจำนวนชิ้นเท่ากัน ให้เลือกสินค้าที่มีรหัสสินค้ามาก่อนตามพจนานุกรม) มาลงโฆษณาในสื่อต่าง ๆ จงเขียนฟังก์ชัน vector<string> getPromotion(vector <pair<string, double> > price, vector <pair<string, int > > sale, vector <pair<double, double> > query, int N, int S) เพื่อคำนวณหาสินค้าที่ต้องทำโปรโมชัน ฟังก์ชันนี้จะคืนค่า vector ของรหัสสินค้าที่ควรทำโปรโมชัน โดยรหัสสินค้าที่คืนมานั้นจะต้องไม่ซ้ำกันเลย ขอให้พยายามทำให้ได้ประสิทธิภาพในการทำงานที่เร็วที่สุด **(ให้ระวังว่าสินค้าที่ขายได้มากนั้นอาจจะเป็นสินค้าที่ถูกค้นหามากก็เป็นได้ หรือ จำนวนสินค้าที่มีอาจจะมีน้อยกว่า N หรือ S ก็เป็นได้ นอกจากนี้ การค้นหาราคาสินค้าโดยผู้ใช้นั้นช่วงราคาที่ถามอาจจะไม่มีสินค้าอยู่ก็เป็นได้)**

```
//ถ้าต้องการเขียนฟังก์ชันเพิ่มเติมสามารถทำได้ที่ตรงนี้









vector<string> getPromotion(vector<pair<string,double> > price,
                            vector<pair<string, int> > sale,
                            vector<pair< double, double> > query, size_t N, size_t S) {

}
```

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 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<value_type>

Capacity	`void resize(size_t n);  // make the size of the vector be exactly n`
Element Access	`operator[] (size_t n);`

Modifier	`void push_back(const value_type &val);`
	`iterator insert(iterator position, const value_type &val);`
	`iterator insert(iterator position, InputIterator first, InputIterator last);`
	`iterator erase(iterator position);`

Class set<value_type>

Operation	`iterator find (const key_type& val);` `size_type count (const key_type& val);`

Modifier	`pair<iterator,bool> insert (const value_type& val);`
	`void insert (InputIterator first, InputIterator last);`
	`void erase (iterator position);`
	`size_type erase (const value_type& val);`
	`void erase (iterator first, iterator last);`

Class map<key_type,mapped_type>

Element Access	`mapped_type& operator[] (const key_type& k);`
Operation	`iterator find (const key_type& val);` `size_type count (const key_type& val);`

Modifier	`pair<iterator,bool> insert (const pair<key_type,mapped_type>& val);`
	`void insert (InputIterator first, InputIterator last);`
	`void erase (iterator position);`
	`size_type erase (const key_type& val);`
	`void erase (iterator first, iterator last);`

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 value_type& val); // add the element`
	`void pop();  // remove the element`

Class queue<value_type>

Element Access	`value_type front();` `value_type back();`

Class stack<value_type>

Element Access	`value_type top();`

Class priority_queue<value_type>

Element Access	`value_type top();`

Useful function

```
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);
```