FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introduction to Data Structures
Year 2nd , First Semester , Midterm Examination 11 Oct 2019 8:30-11:30

ชื่อ-นามสกุล_____เลขประจำตัว_____ CR61___

หมายเหตุ

1. ข้อสอบมีทั้งหมด 11 ข้อในกระดาษคำถามคำตอบจำนวน 13 แผ่น 13 หน้า คะแนนเต็ม 106 คะแนน
2. ไม่อนุญาตให้นำตำราและเครื่องคำนวณต่างๆ ใดๆ เข้าห้องสอบ
3. ควรเขียนตอบด้วยลายมือที่อ่านง่ายและชัดเจน สามารถใช้ดินสอเขียนคำตอบได้
4. ห้ามการหยิบยืมสิ่งใดๆ ทั้งสิ้น จากผู้สอบอื่นๆ เว้นแต่ผู้คุมสอบจะหยิบยืมให้
5. ห้ามนำส่วนใดส่วนหนึ่งของข้อสอบออกจากห้องสอบ ข้อสอบเป็นทรัพย์สินของราชการซึ่งผู้ลักพาอาจมีโทษทางคดีอาญา
6. ผู้เข้าสอบสามารถออกจากห้องสองได้ หลังจากผ่านการสอบไปแล้ว 45 นาที
7. เมื่อหมดเวลาสอบ ผู้เข้าสอบต้องหยุดการเขียนใดๆ ทั้งสิ้น
8. ผู้ที่ปฏิบัติเข้าข่ายทุจริตในการสอบ ตามประกาศคณะวิศวกรรมศาสตร์
   มีโทษ คือ ได้รับ สัญลักษณ์ F ในรายวิชาที่ทุจริต และพักการศึกษาอย่างน้อย 1 ภาคการศึกษา

ห้ามนิสิตพกโทรศัพท์และอุปกรณ์สื่อสารไว้กับตัวระหว่างสอบ หากตรวจพบจะถือว่านิสิตกระทำผิดเกี่ยวกับการสอบ อาจต้องพ้นสภาพการเป็นนิสิต หรือ ให้ได้รับ F และอาจพิจารณาให้ถอนรายวิชาอื่นทั้งหมดที่ลงทะเบียนไว้ในภาคการศึกษานี้

*** ร่วมรณรงค์การกระทำผิด หรือทุจริตการสอบเป็นศูนย์ที่คณะวิศวกรรมศาสตร์ ***

ข้าพเจ้ายอมรับในข้อกำหนดที่กล่าวมานี้ ข้าพเจ้าเป็นผู้ทำข้อสอบนี้ด้วยตนเองโดยมิได้รับการช่วยเหลือหรือให้ความช่วยเหลือ ในการทำข้อสอบนี้

ลงชื่อนิสิต.....
วันที่.....

หมายเหตุเพิ่มเติม:

1. **หากที่ไม่พอเขียนคำตอบให้เขียนในด้านหลังได้แต่ต้องระบุไว้ให้ชัดเจน**
2. **ให้เขียนชื่อ-นามสกุล รหัสนิสิต และ** CR **ให้ครบทุกหน้า**
3. **ในข้อที่มีการเขียนโปรแกรมนั้น จะมีการพิจารณาประสิทธิภาพในการทำงานของโปรแกรมที่เขียนมาด้วย ให้พยายามเขียนโปรแกรมที่ทำงานได้เร็วที่สุด**

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

1. (8 คะแนน) จงเขียนส่วนของโปรแกรมที่ทำการพิมพ์เฉพาะข้อมูลประเภท int ทั้งหมดจากตัวแปร x ซึ่งถูกประกาศเป็นประเภทดังต่อไปนี้

```
1.1  vector<pair<int,string>> x;



1.2  queue<vector<int>> x;



1.3  map<int,pair<string,int>> x;



1.4  vector<map<vector<string>,stack<int>>> x;
```

2. (5 คะแนน) จงวิเคราะห์อัตราการเติบโตของส่วนของโปรแกรมต่อไปนี้ ให้ตรงกับความเป็นจริงมากที่สุดโดยให้เขียนอยู่ในรูปสัญกรณ์เชิงเส้นกำกับของตัวแปร n ตามที่กำหนดให้ในโจทย์แต่ละข้อ

2.1 ```// n is some positive integer for (int i = 0;i < n;i++) { pair<int,int> p = {1,2}; cout << p.first << endl; }```	**Θ**(___)
2.2 ```// v is a vector<int> of n elements queue<int> q; for (auto &x : v) { q.push(x); }```	**Θ**(___)
2.3 ```// v is a vector<int> of n elements stack<vector<int>> s; for (int i = 0;i < n;i++) { s.push(v); }```	**Θ**(___)
2.4 ```// m is a map<int,int> of n elements vector<int> v; for auto(&x : m) { v.insert(v.begin(),x.first); }```	**Θ**(___)
2.5 ```// v is a vector of n elements while (v.empty() == false) { v.erase(v.end()-1); }```	**Θ**(___)

3. (5 คะแนน) จงเติมส่วนของโปรแกรมลงในที่ว่างด้านล่างนี้เท่านั้น เพื่อให้โปรแกรมทำงานโดยมีอัตราการเติบโตเป็นดังที่ระบุ

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

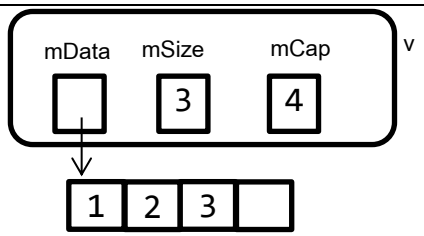
3.1 $\Theta(n)$	`vector<int> v;` `for (int i = 0;i < n;i++) {` `  v.insert(_____,i);` `}`
3.2 $\Theta(n^2)$	`vector<int> v(n);` `for (int i = 0;i < n;i++) v[i] = i;` `for (int i = 0;i < n;i++) {` `  find(v.begin(), v.end(), _____);` `}`
3.3 $O(n)$	`priority_queue<int> pq;` `for (int i = 0;i < n;i++) {` `  pq.push(i);` `  pq._____;` `}`
3.4 $O(n)$	`map<int,int> m;` `for (int i = 0;i < n;i++) {` `  m[_____] = _____;` `}`
3.5 $O(n \log n)$	`set<int> s;` `for (int i = 0;i < n;i++) s.insert(i);` `for (int i = 0;i < n;i++) {` `  s._____` `}`

4. (8 คะแนน) จงเติมส่วนของโปรแกรมลงในที่ว่างด้านล่างนี้เท่านั้นเพื่อให้โปรแกรมทำงานตามที่โจทย์ระบุ

4.1 ลบข้อมูลตัวที่ 2 ออกจาก vector<int> v	`v.erase(_____);`
4.2 พิมพ์คำว่า “YES” ก็ต่อเมื่อมีค่า x อยู่ใน vector<int> v ตำแหน่งใด ๆ ก็ได้ ยกเว้นตำแหน่งแรกสุด	`if (find(_____,v.end(), x) == _____) {` `  cout << "NO" << endl;` `} else {` `  cout << "YES" << endl;` `}`
4.3 คำนวณผลรวมทั้งหมดของข้อมูลใน set<int> s ให้อยู่ในตัวแปร sum	`int sum  = 0;` `for (_____) {` `  sum += x;` `}`
4.4 ทำให้ queue มีข้อมูลเหลือไม่มากกว่า 3 ตัว	`while (_____) {`

	q.pop(); }
4.5 คืนค่าที่น้อยที่สุดใน priority_queue<int> pq	int getmin(priority_queue<int> pq) { while (pq.size() > _____) pq.pop(); return _____ }
4.6 ลบข้อมูลตัวที่มี Key ที่มีค่าน้อยสุดออกจาก map<int,int> m	m._____

5. (5 คะแนน) จงวาดแผนภูมิเพื่อแสดงลักษณะและค่าของ data member ในโครงสร้างข้อมูล หลังจากที่โปรแกรมได้ทำงานจนจบคำสั่งสุดท้ายในแต่ละข้อดังต่อไปนี้

5.0	vector<int> v; v.push_back(1); v.push_back(2); v.push_back(3); // ตัวอย่าง	
5.1	vector<int> v; v.resize(5);	
5.2	vector<int> v; for (int i = 0;i < 8;i++) v.push_back(i); for (int i = 0;i < 4;i++) v.erase(v.end()-1);	
5.3	stack<int> s; for (int i = 0;i < 4;i++) s.push(i*10); s.pop(); s.pop();	
5.4	queue<int> q; for (int i=0;i<3;i++) q.push(i); for (int i=0;i<2;i++) q.pop(); for (int i=0;i<3;i++) q.push(10);	
5.5	vector<pair<int,int>> v; for (int i = 0;i < 4;i++) { v.push_back({i,-i}); }	

6. (10 คะแนน) โดยปรกติแล้ว เราไม่สามารถเปรียบเทียบ CP::stack สองตัวใด ๆ ได้ว่าใครมากว่าหรือน้อยกว่ากัน ในโจทย์ข้อนี้ ให้นิสิตเพิ่มบริการให้กับคลาส CP::stack เพื่อให้สามารถเปรียบเทียบกันได้ โดยให้เขียน operator< ของ CP::stack โดยมีหลักการเปรียบเทียบ CP::Stack<T> สองตัว s1 กับ s2 ดังนี้ กำหนดให้ a และ b เป็นข้อมูลตัวที่อยู่บนสุด (top of stack) ของ stack s1 และ s2 ตามลำดับ เราจะแยกพิจารณาเป็นกรณีต่าง ๆ ดังนี้ กรณีที่ 1: ถ้าหากว่าทั้งสอง stack ไม่มีข้อมูลตัวดังกล่าวอยู่ (คือเป็น stack ว่างทั้งคู่) เราจะถือว่าทั้งสอง stack มีค่าเท่ากัน กรณีที่ 2: หากว่า stack ใด stack หนึ่ง มีข้อมูลตัวดังกล่าว แต่อีก stack ไม่มีข้อมูล เราจะ

ถือว่า stack ตัวที่ไม่มีข้อมูลนั้น น้อยกว่า stack ตัวที่มีข้อมูล กรณีที่ 3: เมื่อทั้ง s1 และ s2 มีข้อมูลตัวดังกล่าว หาก a < b เราจะสรุปว่า s1 น้อยกว่า s2 แต่ถ้าหาก a > b เราจะสรุปว่า s2 น้อยกว่า s1 สุดท้าย หาก a == b แล้ว เราจะกำหนดว่า s1 < s2 ก็ต่อเมื่อ s1' < s2' เท่านั้น โดยที่ s1' คือ s1 ที่ทำการ pop ข้อมูลทิ้งไปหนึ่งตัว และ s2' คือ s2 ที่ทำการ pop ข้อมูลทิ้งไปหนึ่งตัวเช่นกัน

ตัวอย่างเช่น สมมติให้ s1 มีข้อมูลเป็น {1,2,3} และ s2 มีข้อมูลเป็น {2,2,3} โดยที่ top of stack คือข้อมูลตัวขวาสุด เราจะสรุปได้ว่า s1 น้อยกว่า s2 แต่หากว่า s1 = {1,2} และ s2 = {2} เราจะถือว่า s2 น้อยกว่า s1 เป็นต้น

```
template <typename T>
class stack {
  protected:
    T *mData; size_t mCap, mSize  // ห้ามประกาศ data member เพิ่มเติม
  public:
    // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง โดยฟังก์ชันอื่น ๆ มิได้ระบุไว้เพื่อประหยัดหน้ากระดาษ ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
    // ให้เขียนเติมฟังก์ชันด้านล่างนี้ ให้ทำงานให้ถูกต้อง
    bool operator<(const other &s2) {









    }
}
```

7. (10 คะแนน) จงเพิ่มบริการให้กับคลาส CP::vector โดยให้เขียน constructor เพิ่มเติม คือ vector(const queue<T> &q) ซึ่งจะสร้าง vector ที่มีข้อมูลประกอบด้วยข้อมูลภายใน queue นั้นทั้งหมดโดยให้เรียงลำดับข้อมูลจากท้ายคิว กล่าวคือ ข้อมูลตัวที่อยู่ท้ายคิวจะต้องอยู่ที่ช่อง 0, ข้อมูลตัวก่อนท้ายคิวอยู่ช่องที่ 1,..., ข้อมูลที่หัวคิวจะอยู่ช่องสุดท้ายใน vector ให้สังเกตว่า q นั้นได้รับมาเป็นแบบ const ดังนั้นเราไม่สามารถทำการแก้ไข q ได้

```
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
    vector(const queue<T> &q) {









    }
};
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

8. (15 คะแนน) โจทย์ข้อนี้เป็นการเขียนคลาสสำหรับระบบประมูลขายของแบบออนไลน์บนเว็บไซต์หนึ่ง การประมูลแบบออนไลน์เป็นดังนี้ กำหนดให้มีผู้ใช้ระบบอยู่หลายคน แต่ละคนมีชื่อที่ไม่ซ้ำกันเลย และผู้ใช้เหล่านั้นมีสินค้าที่ต้องการนำมาประมูล สินค้าแต่ละชนิดมีชื่อที่ไม่ซ้ำกันเลย และผู้ใช้ที่ทำการประมูลสินค้าชนิดนั้นอาจจะเปิดประมูลสินค้าชนิดเดียวกันพร้อมกันหลาย ๆ ชิ้นก็ได้ (เช่น เปิดประมูล iPhone 10 เครื่องพร้อมกัน)

   การประมูลเริ่มขึ้นเมื่อเจ้าของสินค้าทำการลงทะเบียนชื่อสินค้าพร้อมด้วยจำนวนชิ้นที่ต้องการขาย เมื่อลงทะเบียนแล้ว ผู้ใช้แต่ละคนสามารถทำการประมูลสินค้า โดยทำการ “เสนอราคาซื้อ” ของสินค้าดังกล่าว ถ้าหากผู้ใช้ต้องการซื้อมากกว่าหนึ่งชิ้น ก็สามารถทำการเสนอราคาซื้อ ของชื่อเดียวกันมากกว่าหนึ่งครั้งก็เป็นได้

   เมื่อเวลาผ่านไปตราบเท่าที่ผู้ขายพอใจแล้ว ผู้ขายสามารถปิดการประมูล โดยเมื่อปิดแล้วระบบจะให้สิทธิ์ในการซื้อสินค้าชนิดดังกล่าวกับผู้ที่ให้ราคามากที่สุด ถ้ามีผู้ให้ราคามากสุดเท่ากัน ก็จะให้สิทธิ์กับผู้ที่เสนอราคาประมูลก่อน โดยผู้เสนอราคาที่ได้สิทธิ์นั้น ก็จะสามารถเลือกที่จะใช้สิทธิ์หรือไม่ใช้ก็ได้ ถ้าหากใช้สิทธิ์ จำนวนสินค้านั้นก็จะลดลงไปหนึ่งหน่วยเนื่องจากได้ขายให้ผู้เสนอราคารายนั้นไป แต่ถ้าหากไม่ใช้สิทธิ์ จำนวนสินค้าก็จะมีเท่าเดิม หลังจากนั้นระบบจะให้สิทธิ์กับผู้ประมูลรายถัดไปเรียงตามลำดับเช่นเดิม จนกระทั่งสินค้าหมดหรือจนกระทั่งไม่มีผู้ได้รับสิทธิ์แล้ว ตัวอย่างเช่น หากเราประมูล iPhone 2 เครื่อง และมีผู้เสนอราคาเรียงตามเวลาเป็นดังนี้ สมชายเสนอ 5 บาท 1 ครั้ง และ 10 บาท อีกหนึ่งครั้ง หลังจากนั้น สมศักดิ์เสนอ 10 บาท 1 ครั้ง แล้ว สมหญิงเสนอ 12 บาทหนึ่งครั้ง เมื่อปิดประมูล ระบบจะให้สิทธิ์ตามลำดับเป็นดังนี้ สมหญิง (12 บาท) --> สมชาย (10 บาท) --> สมศักดิ์ (10 บาท) --> สมชาย (5 บาท)

   จงเขียนคลาส MultiAuction ซึ่งต้องมีบริการ (public member function) ดังต่อไปนี้

- void open_auction(string item, int qty) เป็นการเปิดประมูลสินค้าชื่อ item จำนวน qty ชิ้น (รับประกันว่า การเรียก open_auction แต่ละครั้งนั้น item จะไม่ซ้ำกันเลย และ qty มีค่าเป็นบวกเสมอ)
- void add_bidding(string user, string item, int price) เป็นการระบุว่าผู้ใช้ชื่อ user ทำการเสนอราคาซื้อสินค้าชื่อ item ด้วยราคา price (รับประกันว่า item นั้นจะเป็น item ที่มีการเปิดประมูลแล้วและยังไม่ปิดการประมูลเท่านั้น)
- vector<pair<string,int>> close_auction(string item) เป็นการปิดการประมูล โดยระบบจะต้องคืนรายการของผู้ใช้ที่ยืนยันใช้สิทธิ์ในการซื้อของดังกล่าว โดยมีการสอบถามไปยังผู้ใช้ในลำดับที่ถูกต้อง และรายการที่คืนมานั้นจะต้องเรียงลำดับอย่างถูกต้องเช่นกัน

  ในฟังก์ชันนี้ นิสิตสามารถเรียกใช้ฟังก์ชัน bool ask_user_confirmation(string user, string item, integer price) ซึ่งเป็นฟังก์ชันที่จะทำการถามไปยังผู้ใช้ชื่อ user ที่ได้ประมูล item ด้วยราคา price ไว้ว่า จะยืนยันการซื้อสินค้าดังกล่าวหรือไม่ ถ้าหากผู้ใช้ยืนยัน ฟังก์ชันนี้ก็จะคืนค่า true มาให้

  จากตัวอย่างการประมูล iPhone ด้านบนจำนวนสองเครื่อง การเรียก close_auction(“iPhone”) นั้น จะมีการถามไปยัง user ตามลำดับข้างต้น สมมติว่าเมื่อถามสมหญิงด้วยราคา 12 บาท และ เมื่อถามสมชายด้วยราคา 10 บาท ทั้งคู่ต่างปฏิเสธไม่ใช้สิทธิ์ แต่เมื่อถามไปยังสมศักดิ์ด้วยราคา 10 บาท และ สมชายด้วยราคา 5 บาทแล้ว ทั้งคู่ยืนยัน ฟังก์ชันนี้ควรจะคืนค่าเป็น { {“สมศักดิ์”, 10}, {“สมชาย”,5} } เป็นต้น

ในโจทย์ข้อนี้ การทำงานนั้นจะต้องถูกต้อง และ รวดเร็ว แต่ถ้าหากทำให้รวดเร็วไม่ได้ ก็ขอให้พยายามทำให้ถูกต้องตามข้อกำหนดมากที่สุด ให้ออกแบบคลาสโดยคำนึงว่าจำนวนชนิดของสินค้า และจำนวนชิ้นในการประมูล และผู้ใช้ นั้นมีมากมาย และมีการเสนอราคาและยกเลิกการเสนอราคาบ่อยครั้งมาก ๆ

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

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

8.2 จงเขียนคลาส MultiAuction ตามข้อกำหนดที่ได้ระบุไว้ข้างต้น ถ้าหากเนื้อที่ไม่พอเขียน ให้เขียนไว้ด้านหลังของหน้า 7 เท่านั้น

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

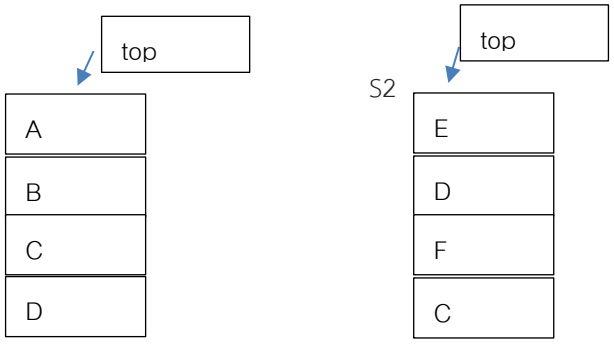
9. (7 คะแนน) ในข้อนี้ให้นิสิตเพิ่มบริการ ของ MyClass::stack (ซึ่งไม่รู้ว่าใช้อะไรเขียน แต่รู้ว่ามีฟังก์ชั่นอะไรบ้าง) โดยให้เพิ่มฟังก์ชันต่อไปนี้

stack<T> union(stack<T> s2)

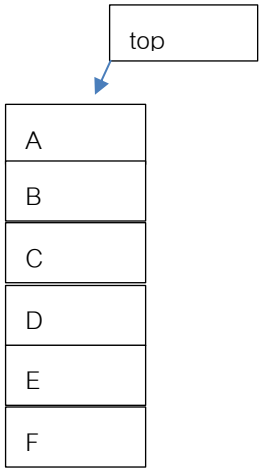
ฟังก์ชั่นนี้ทำการรวมของใน s2 เข้ากับของในสแตกที่เรียกฟังก์ชันนี้ใช้ เกิดผลลัพธ์เป็นสแตกใหม่ โดยไม่เติมของซ้ำ

- ให้สมมุติได้เลย ว่า สแตกแต่ละสแตกนั้น ไม่มีของซ้ำ (แต่ของในสแตกคนละสแตกมีซ้ำกันได้)
- ของในสแตกที่เรียกฟังก์ชันนี้ จะต้องอยู่ด้านบนของสแตกผลลัพธ์
- ส่วนของที่มาจาก s2 จะต้องอยู่ด้านล่างของสแตกผลลัพธ์

ตัวอย่างเช่น ถ้าสแตกทั้งสองมีของดังนี้



สแตกผลลัพธ์จะได้ดังนี้




```
template <typename T>
class stack {
  protected:
    // ไม่รู้ว่าตัวแปรมีอะไรบ้าง
  public:
     stack()                      { ... }  //ให้ถือว่าโค้ดที่อยู่ใน {…} นั้น มีมาให้เรียบร้อยแล้ว
    stack(const stack<T>& a)      { ... }
    ~stack()                      { ... }

    bool   empty()                { ... }
    size_t size()                 { ... }
    //----------------- access -----------------
    const T& top()                { ... }
    //----------------- modifier ---------------
    void push(const T& e)         { ... }
    void pop()
```

```
    stack<T> union(stack<T> s2) {
      // เติม C++ code ตรงนี้  แบ่งครึ่งหน้าเพื่อเพิ่มพื้นที่ได้นะ









    }
}
```

10. (5 คะแนน) ในข้อนี้ให้นิสิตเพิ่มบริการ ของ CP::vector โดยให้เพิ่มฟังก์ชันต่อไปนี้

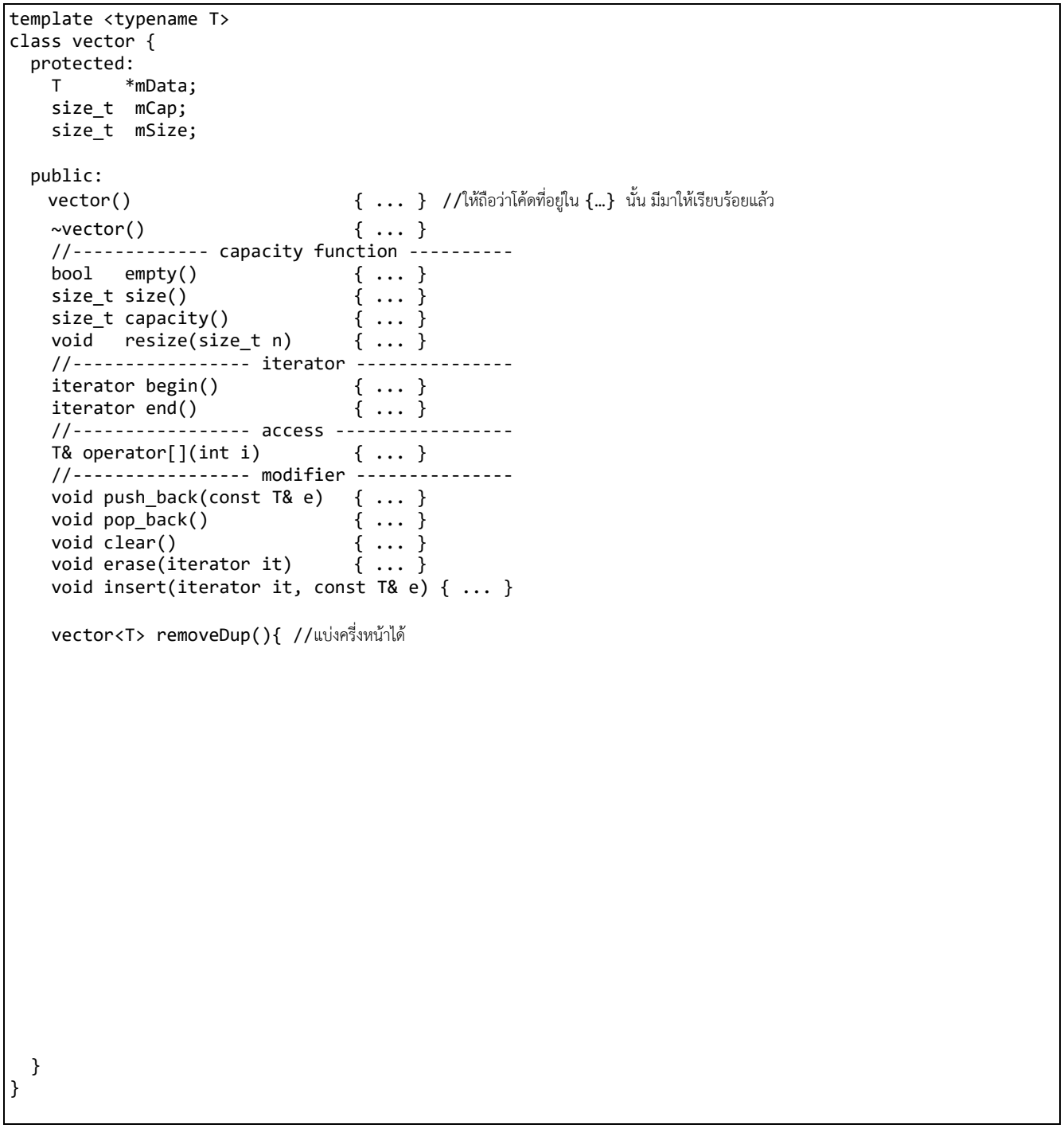
vector<T> removeDup()

ฟังก์ชั่นนี้ทำการเอาของที่ซ้ำ ในเวคเตอร์ ออก จนของแต่ละอย่างเหลือแค่หนึ่งชิ้น รีเทิร์นเวคเตอร์ใหม่ที่เกิดจากการเอาของซ้ำทิ้งไปแล้ว อีกนัยหนึ่งคือ สร้างเซตขึ้นมา นั่นเองโดยห้ามนิสิตใช้ คลาสหรือฟังก์ชั่นของ STL ใดๆ

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

  public:
    vector()                    { ... } //ให้ถือว่าโค้ดที่อยู่ใน {…} นั้น มีมาให้เรียบร้อยแล้ว
    ~vector()                   { ... }
    //------------- capacity function ----------
    bool   empty()              { ... }
    size_t size()               { ... }
    size_t capacity()           { ... }
    void   resize(size_t n)     { ... }
    //----------------- iterator ---------------
    iterator begin()            { ... }
    iterator end()              { ... }
    //----------------- access -----------------
    T& operator[](int i)        { ... }
    //----------------- modifier ---------------
    void push_back(const T& e)  { ... }
    void pop_back()             { ... }
    void clear()                { ... }
    void erase(iterator it)     { ... }
    void insert(iterator it, const T& e) { ... }

    vector<T> removeDup(){ //แบ่งครึ่งหน้าได้
```



```
  }
}
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

11. (8 คะแนน) มีความต้องการให้นิสิตเขียน priority queue ขึ้นมาเอง โดยสร้างจาก

vector<queue<pair<T,int>>> // พวกนี้มาจาก std ทั้งหมด

โดยมีข้อกำหนดดังนี้

- ข้อมูลที่เก็บ นั้น เก็บเป็น pair
- ค่า int ใน pair คือค่า priority โดยให้มีค่าตั้งแต่ 0 ขึ้นไป
- ของที่อยู่ในคิวเดียวกันจะมีค่า priority เท่ากัน
- v[i] จะเก็บคิวที่ของในคิวนั้นเป็น pair ที่มีค่า priority เป็น i
- คิว สามารถเป็นคิวว่างได้ คิวว่างจะไม่โดนลบออกจากเวคเตอร์ของเรา
- ค่าที่สำคัญที่สุด คือ pair ที่ค่า priority น้อยสุด ที่เจอเป็น pair แรก เมื่อทำการค้นหาในเวคเตอร์

```
template <typename T>
class priority_queue {
protected:
     vector<queue<pair<T,int>>>   v;
public:
     priority_queue()          { ... } //ให้ถือว่าโค้ดที่อยู่ใน {…} นั้น มีมาให้เรียบร้อยแล้ว
    ~priority_queue()    { ... }
     //------------- capacity function ----------
     bool   empty() const {




     }
     size_t size()  const {




     }
     //----------------- access -----------------
     const T& top() const {









     }
     //----------------- modifier ---------------
     void push(const T& e) {









     }
     void pop() {









     }
}
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

12. (10 คะแนน) จงเพิ่มเมท็อด `insertAt` ให้กับ `CP::queue` เพื่อนำข้อมูลจากแถวคอย q มาแทรกยังตำแหน่งที่ `k` จากหัวคิวนี้ (หัวคิวคือ `k = 0`) นิสิตสามารถเรียกใช้บริการต่างๆที่มีอยู่แล้วใน `CP::queue` หรือเข้าถึง member variable ใดๆใน CP::queue ก็ได้

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

    void insertAt(int k, const CP::queue<T>& q) {









        }
};
```

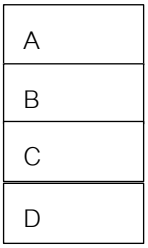
```
queue(const queue<T>& a) {...}
queue() {...}
queue<T>& operator=(queue<T> other) {...}
~queue() {...}
bool     empty() const {...}
size_t   size() const {...}
const T& front() const {...}
const T& back() const {...}
void     push(const T& element) {...}
void     pop() {...}
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

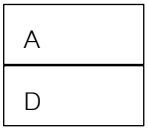
13. (10 คะแนน) จงเพิ่มเมท็อด removeAtK ให้กับ `CP::stack` เพื่อนำข้อมูล m ตัวออกจากกองซ้อนโดยเริ่มตั้งแต่ตำแหน่งที่ `k` จากบนสุดของกองซ้อน (บนสุดของกองซ้อนคือ `k = 0`)

ตัวอย่าง

S



S หลังเรียก S.removeAt(1,2)



นิสิตสามารถเรียกใช้บริการต่างๆที่มีอยู่แล้วใน `CP::stack` หรือเข้าถึง member variable ใดๆใน CP::stack ก็ได้

โดยการจัดเก็บข้อมูลของ CP::stack นั้นเหมือนกับที่เรียนในคาบ

```
stack(const stack<T>& a) {...}
stack() {...}
stack<T>& operator=(stack<T> other) {...}
~stack() {...}
bool     empty() const {...}
size_t   size() const {...}
const T& top() const {...}
void     push(const T& element) {...}
void     pop() {...}
```

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

    void removeAt(int k, int m) {









        }
};
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

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

Class 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 ที่ใช้ได้เฉพาะ list	void push_front(const ValueT& val); void pop_font; void remove(const ValueT& val);

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);
```