FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2110211 Introduction to Data Structures

YEAR II, Second Semester, Midterm Examination, March 4, 2016, Time 13:00 – 16:00

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 11 ข้อในกระดาษคำถามคำตอบจำนวน 9 แผ่น 9 หน้า คะแนนเต็ม 87 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. (12 คะแนน) จงตั้งชื่อฟังก์ชัน (ที่ได้ความหมายเดียวกับที่ทำงาน) พร้อมทั้งเขียนอธิบายเพิ่มเติมอีกเล็กน้อยได้ในช่องว่างด้านขวา และเขียนผลลัพธ์ของการทำฟังก์ชันที่มี parameter ตามที่แสดงด้านขวา

```
void erase middle element (vector<int>& v ) {
  v.erase( v.begin() + v.size()/2 );
}
```

ตัวอย่าง

```
void _____(vector<int> & v, int k) {
    auto itr = v.begin();
    for(int i=0; i<k; i++) {
        itr = v.insert(itr,v.back());
        v.pop_back();
    }
}
```

ถ้า **v = <3,4,1,5,4>, k=2**

ผลลัพธ์ **v =**_____

```
void _____(vector<int>& v) {
    map<int,int> m;
    for (auto x : v) m[x]++;
    auto itr = v.begin();
    for (auto x : m) {
        for (int i=0; i<x.second; i++) {
            *itr = x.first;
            itr++;
        }
    }
}
```

ถ้า **v = <3,4,1,5,6>**

ผลลัพธ์ **v =**_____

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

```
int _____(vector<int> &v, int k) {
    // k = 1,2,3, ..., v.size()
    priority_queue<int> pq;
    for (auto x : v) pq.push(-x);
    for (int i=1; i<k; i++) pq.pop();
    return -pq.top();
}
```

ถ้า **v = <3,4,1,5,4>, k=2**

ผลลัพธ์ _____

```
void _____(queue<int> &q) {
    map<int,int> m;
    for(int i=0,n=q.size(); i<n; i++) {
        int x = q.front();
        q.pop();
        if (m[x]==0) {
            m[x]++;
            q.push(x);
        }
    }
}
```

ถ้า **q = [3,4,1,5,4]** (front)

ผลลัพธ์ **q =** _____

2. (6 คะแนน) จงเขียนประเภทข้อมูลของ parameters ในช่องว่างที่เว้นว่างไว้ของฟังก์ชัน ที่ทำให้การทำงานของฟังก์ชันเป็นไปตามชื่อฟังก์ชัน

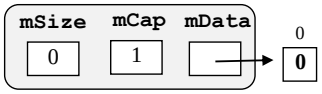
```
float sum( _____ d ) {
    float sum = 0.0;
    for(size_t i=0; i<d.size(); i++) {
        for (size_t j=0; j<d[i].size(); j++) {
            sum += d[i][j];
        }
    }
    return sum;
}
```

```
float sum( _____ d ) {
    float sum = 0.0;
    for (auto r : d) {
        for (auto c : d[r.first]) {
            sum += d[r.first][c.first];
        }
    }
    return sum;
}
```

```
float sum( _____ d ) {
    float sum = 0;
    auto i = d.begin();
    while (i != d.end()) {
        sum += *i; i++;
    }
    return sum;
}
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

3. (6 คะแนน) จงวาดรูปโครงสร้างข้อมูลในช่องทางขวา หลังจากทำคำสั่งทั้งหมดในช่องทางซ้าย (ใช้คลาสที่เขียนในชั้นเรียน)

ตัวอย่าง `CP::vector<int> v;`	 **ตัวอย่าง**

`CP::vector<int> v;`
`for (int i=0; i<5; i++)`
`  v.push_back(i);`	

`CP::vector<int> v;`
`for (int i=0; i<5; i++)`
`  v.push_back(i);`
`v.insert(v.begin()+4,99);`
`v.erase(v.end()-5);`	

`CP::queue<int> q;`
`for (int i=0; i<7; i++) q.push(i);`
`for (int i=0; i<5; i++) q.pop();`
`for (int i=7; i<10; i++) q.push(i);`	

********** สำคัญ!!! **********

ตั้งแต่ข้อ 4 เป็นต้นไปเป็นการเขียนโปรแกรม การเขียนโปรแกรมจะต้องไม่ผิด syntax ในส่วนที่สำคัญ โดยเฉพาะอย่างยิ่งการเรียก function และ argument ของฟังก์ชันจะต้องถูกต้อง นอกจากนี้คะแนนจะแปรผันตาม "ประสิทธิภาพ" ของโปรแกรมที่เขียนมา **** ให้สังเกตถึงข้อกำหนดต่าง ๆ ในโจทย์ให้รอบคอบ <u>การไม่ทำตามข้อกำหนดจะมีผลต่อคะแนนเป็นอย่างมาก</u> ******

4. (5 คะแนน) ให้ v1 และ v2 เป็น vector จงเขียนฟังก์ชั่น

vector<T> alternate(const vector<T>& v1, const vector<T>& v2)

เพื่อคืนค่า vector ที่เกิดจากการที่นำข้อมูลแต่ละตัวใน v1 มาใส่ตามลำดับสลับกับข้อมูลใน v2 ตามลำดับไปเรื่อยๆจนใช้ข้อมูลใน vector ตัวใดตัวหนึ่งหมดจากนั้นก็นำข้อมูลที่เหลือใน vector อีกอันหนึ่งมาใส่ โดยที่ห้ามแก้ไขข้อมูลใน v1 และ v2 ตัวอย่างเช่น หาก v1=<1,2,3,4> v2=<8,9,10,11,12,13> v3=alternate(v1,v2); v3 จะมีค่าเป็น <1,8,2,9,3,10,4,11,12,13>

```
template <typename T>
vector<T> alternate(const vector<T>& v1, const vector<T>& v2) {
    // เติมโค้ดที่นี่









}
```

5. (5 คะแนน) ในการใช้งานเมทริกซ์ (Matrix) ทางวิศวกรรมนั้น ในหลายโอกาสนั้น ช่องส่วนมากของเมทริกซ์ จะมีค่าเป็น 0 ซึ่งเราเรียกเมทริกซ์ประเภทนี้ว่าเมทริกซ์มากเลขศูนย์ (Sparse Matrix) ซึ่งหากเราเก็บค่าของช่องต่างๆของเมทริกซ์ตรงๆ ก็จะใช้เนื้อที่เปลืองมาก เราจึง

จะเก็บเมทริกซ์มากเลขศูนย์โดยการระบุว่าแถว และ คอลัมป์ ไหนมีค่าอะไรแทน ด้วย vector<pair< pair<int,int>, float> > โดยที่ pair<int,int> เก็บแถวและคอลัมป์ของช่อง และ ค่าของช่องนั้นแทนด้วย float (ที่ไม่ใช่ 0) โดยจะถือว่าช่องที่เหลือมีค่าเป็น 0

ตัวอย่างเช่น เมทริกซ์ $M = \begin{bmatrix} 3.2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 7.4 & 0 \end{bmatrix}$ แทนได้โดย < ((0,0),3.2) , ((2,1),7.4) >

ในข้อนี้ให้นิสิตเขียน ฟังก์ชั่นที่คืนเมทริกซ์มากเลขศูนย์ ที่เกิดจากการบวกเมทริกซ์มากเลขศูนย์สองอันเข้าด้วยกัน

typedef vector<pair< pair<int,int>, float> > SparseMat;

SparseMat addSparseMatrix(const SparseMat & M, const SparseMat N);

เช่นให้ M = < ((0,0),3.2) , ((2,1),7.4) > N = < ((0,0),4.1) , ((1,1),1.1) , ((2,1),-7.4), ((3,5),2.3) >

ผลลัพท์ของการบวก O=addSparseMatrix(M,N) คือ O=< ((0,0),7.3) , ((1,1),1.1) , ((3,5),2.3)>

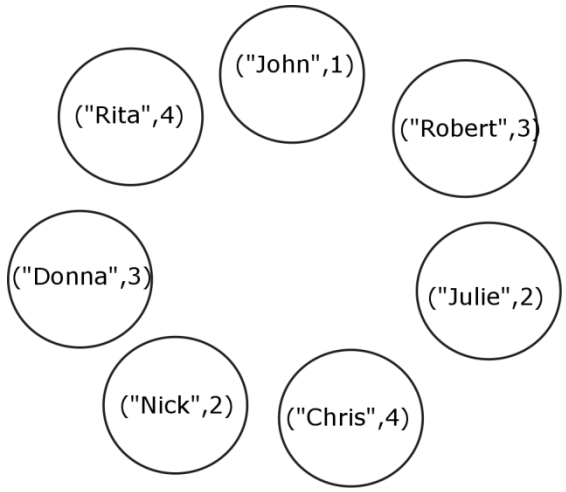
ฟังก์ชั่นนี้สามารถคืนลำดับ ของแถว,คอลัมป์ในลำดับใดก็ได้ แต่จะต้องระบุค่าช่องเพียงครั้งเดียวและช่องที่ระบุจะต้องมีค่าไม่เท่ากับ 0 ให้พยายามทำให้การทำงานนั้นเร็วที่สุด นิสิตสามารถเรียกใช้ class stl อื่นใดก็ได้ (ไม่จำเป็นต้อง #include)

```
typedef vector<pair< pair<int,int>, float> >  SparseMat;
SparseMat addSparseMatrix(const SparseMat &M, const SparseMat &N) {
      // เติมโค้ดที่นี่









}
```

6. (5 คะแนน) เกมส์ส่งต่อ เกมส์นี้จะมีผู้เล่นอยู่หลายคนโดยยืนเรียงกันเป็นวงกลมตามเข็มนาฬิกา(หากมองจากด้านบน) โดยคนที่ i เลือกตัวเลขจำนวนเต็มหนึ่งตัว k ตั้งแต่ 1 ถึง 10 เอาไว้ โดยการเล่นจะคือคนแรกจะได้ลูกบอล จากนั้นเค้าก็จะโยนบอลให้คนที่อยู่ที่ลำดับ k คนถัดไปในวงกลมตามเข็มนาฬิกา โดยเกมส์จะจบเมื่อบอลถูกโยนไป x ครั้ง ผู้ที่ชนะก็คือคนที่มีลูกบอลอยู่ในมือ โดยในข้อนี้ให้นิสิตเขียนโปรแกรม

string whoWin(const queue<pair<string,int>>& p, const int x)

เพื่อคืนค่าว่าคนที่ชนะชื่ออะไรโดยที่ p เก็บลำดับของคนที่ยืนเรียงกันเป็นวงกลมไว้ โดย first เก็บชื่อ และ second เก็บค่า k ที่คนคนนั้นเลือก เช่น p.front().first เก็บชื่อของคนแรก, p.front().second เก็บค่า k ที่คนแรกเลือก โดยแต่ละคนอาจเลือกค่า k ไม่เหมือนกันก็ได้



ตัวอย่างเช่น

โดยคนแรกคือ John หาก x=5 เกมส์จะถูกเล่นดังนี้

ครั้งที่ 1: John ค่าk=1, โยนให้ Robert

ครั้งที่ 2: Robert ค่าk=3, โยนให้ Nick

ครั้งที่ 3: Nick ค่าk=2, โยนให้ Rita

ครั้งที่ 4: Rita ค่าk=4, โยนให้ Chris

ครั้งที่ 5: Chris ค่าk=4, โยนให้ John

หลังจากบอลโดนโยนไป 5 ครั้งลูกบอลมาอยู่ที่ John ดังนั้น John จึงเป็นผู้ชนะ whoWin จึงต้องคืนค่า “John“

Hint: นิสิตจะเห็นว่าใน code ด้านล่างมีการสร้าง q ที่ไม่เป็น const จาก p ที่เป็น const ซึ่งนิสิตสามารถแก้ไข q ได้หากจำเป็น ข้อนี้ห้ามใช้ STL อื่น ๆ นอกเหนือจาก queue และ pair

```
string whoWin(const queue<pair<string,int>>& p, const int x)   {
  queue<pair<string,int>> q(p);
  // เติมโค้ดที่นี่









}
```

7. (10 คะแนน) กำหนดให้มีพนักงานอยู่ N คน พนักงานคนที่ i ได้เงินเดือน s[i] บาท เช้าวันหนึ่ง มีงานที่ต้องทำอยู่ K ชิ้น (K > N เสมอ) หัวหน้าจะแจกงานที่ต้องทำในวันนั้นให้กับพนักงานทีละงาน โดยมีวิธีการดังนี้ หัวหน้าจะแจกงานทีละงาน โดยจะแจกงานชิ้นปัจจุบันให้กับคนที่มีค่าอัตราส่วนระหว่างจำนวนงานที่ได้รับ(รวมถึงงานที่กำลังพิจารณานี้) ต่อเงินเดือนน้อยที่สุดก่อน (หากมีหลายคนเท่ากันจะให้ใครก็ได้) และทำไปเช่นนี้ จนครบ K งาน ให้เขียนโปรแกรมเพื่อหาว่าแต่ละคนจะได้งานกี่ชิ้น **นิสิตสามารถใช้ STL ใด ๆก็ได้**

ตัวอย่างเช่น มี 3 คนเงินเดือน s[0] = 8000, s[1] = 14000, s[2] = 5000 มีงาน 6 งาน

หมายเลขงาน	คนที่ 0		คนที่ 1		คนที่ 2		ให้งานใหม่กับคนที่
	#งาน	(#งาน+1)/ s[0]	#งาน	(#งาน+1)/ s[1]	#งาน	(#งาน+1)/ s[2]	
1	0	1/8000	0	**1/14000 ****	0	1/5000	1
2	0	**1/8000 ****	1	2/14000	0	1/5000	0
3	1	2/8000	1	2/14000	0	**1/5000 ****	2
4	1	2/8000	1	**2/14000 ****	1	2/5000	1
5	1	2/8000	2	**3/14000 ****	1	2/5000	1
6	1	**2/8000 ****	3	4/14000	1	2/5000	0
	2		**3**		**1**		

** คือช่องที่มี ค่าอัตราส่วนระหว่างจำนวนงานที่ได้รับ(รวมถึงงานที่กำลังพิจารณานี้) ต่อเงินเดือนน้อยที่สุด

ดังนั้นคำตอบคือ <2,3,1> ให้คืนค่าเป็น vector<int>โดยที่ข้อมูลชิ้นที่ i ใน vector ก็คือจำนวนงานที่คนที่ i ได้รับ

```
vector<int> getWorkAssignments(const vector<float>& s, int K)
{
  // เติมโค้ดที่นี่









}
```

ชื่อ_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

8. (10 คะแนน) ในข้อนี้มี vector<string> name[i] เก็บชื่อของคน N คนโดยรับประกันว่าชื่อไม่ซ้ำกัน และมี vector<pair<string,string> > parent โดยที่ parent[j].first เป็นพ่อของ parent [j].second ให้นิสิตเขียนโปรแกรมที่พิมพ์รายชื่อของ คนที่เป็นปู่ออกมา และบอกด้วยว่าหลานมีใครบ้าง (โดยถ้าใครไม่มีหลาน ก็ไม่ต้องพิมพ์)

   ตัวอย่างเช่น name = {“John”, “Jack”, “Leo”, “Eric”, “Robert”} father={(“Leo”,”John”), (“John”,”Eric”), (“John”,”Robert”), (“Robert”,”Jack”)}

   Output คือ

   John is a grandfather with the following grandson(s): Jack

   Leo is a grandfather with the following grandson(s): Eric Robert

```
void printGrandfatherGrandson(const vector<string>& name, const
vector<pair<string,string>>& parent) {
  // เติมโค้ดที่นี่









}
```

9. (5 คะแนน) สำหรับคลาส CP::queue<T> นั้น เราไม่สามารถ “ขอดู” ข้อมูลตัวอื่น ๆ นอกจากตัวที่อยู่หัวแถว หรือ ท้ายแถวได้ จงเขียนฟังก์ชันเพิ่มเติมให้กับ class CP::queue คือฟังก์ชัน T get_k(int k) ซึ่งจะคืนค่าข้อมูลที่อยู่ใน queue เป็นลำดับที่ K จากหัวแถว (เช่น ถ้า k เป็น 0 ก็หมายถึงหัวแถว หรือ front() นั่นเอง) ฟังก์ชันนี้จะต้องไม่ทำให้ค่าของสมาชิกใน queue เปลี่ยนแปลง

```
template <typename T>
class queue {
  // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
  protected:    T *mData; size_t mCap;  size_t mSize; size_t  mFront;
  public:
    T get_k(int k)
    {
      // เติมโค้ดที่นี่




    }
};
```

10. (10 คะแนน) ให้นิสิตเขียนฟังก์ชั่น .make_unique() เพิ่มเข้าไปใน CP::vector ที่ทำให้ข้อมูลที่ซ้ำกันใน vector โดนลบไป โดยที่ลบตัวที่ปรากฏขึ้นที่หลังเท่านั้น เช่น <A,B,A,C,A,D,B,X>.make_unique() ต้องได้ <A,B,C,D,X> ห้ามเป็น <B,A,C,D,X> หรือ <A,C,D,B,X> เป็น**ต้น** **<u>โดยห้ามนิสิตเรียกใช้ class หรือฟังก์ชั่นใดๆใน STL หรือ CP ทั้งสิ้น รวมถึงห้ามเรียกฟังก์ชันอื่น ๆ ของ CP::vector ด้วยเช่นกัน</u>**

```
class vector {
  public:    typedef T* iterator;
  protected:    T *mData;    size_t mCap;    size_t mSize;
  // คลาสนี้ทำงานได้ตามปรกติทุกอย่าง ให้นิสิตเขียนบริการเพิ่มเติม ตามข้อกำหนดด้านบน
  void make_unique() {
    // เติมโค้ดที่นี่









  }
};
```

11. (15 คะแนน) โจทย์ข้อนี้เป็นการเขียนคลาสที่จำลองการทำงานของระบบจัดการเพลงแบบออนไลน์ ระบบจัดการเพลงจะทำหน้าที่เก็บเพลงที่ผู้ใช้ใส่เข้ามาในระบบ และรองรับการ playlist ซึ่งเป็นรายการของเพลงที่ผู้ใช้เลือกมาจากเพลงที่ใส่เข้ามาในระบบแล้วมาเรียงต่อกันเป็นลำดับของเพลง playlist แต่ละอันนั้นจะมีชื่อแตกต่างกัน เวลาผู้เล่นฟังเพลงนั้น ผู้เล่นจะฟังเพลงทีละ playlist ดังนั้นระบบจะต้องสามารถคืนค่าข้อมูลเพลงของแต่ละ playlist ได้ นอกจากนี้ เมื่อผู้ใช้ฟังเพลงจบทั้ง playlist แล้ว ระบบจะนับจำนวนครั้งที่เพลงแต่ละเพลงได้เล่น และระบบสามารถคืน รายชื่อของเพลงจำนวน k เพลงที่มีการเล่นสูงสุดได้ด้วย
จงเขียนคลาส MusicLibrary ซึ่งมีบริการต่อไปนี้

- void add_song(const MP3 &mp3, string artist, string title) เป็นการเพิ่มข้อมูลเพลงเข้าไปในระบบ เพลงแต่ละเพลงประกอบด้วยข้อมูลสามส่วน คือ artist (ชื่อศิลปิน) title (ชื่อเพลง) และ data ซึ่งเป็นข้อมูลประเภท MP3 ซึ่งมีขนาดใหญ่มาก (~ 50MB ต่อ 1 เพลง) เพลงแต่ละเพลงจะไม่มีชื่อศิลปินและชื่อเพลงซ้ำกันทั้งคู่พร้อมกัน
- void add_to_playlist(string name, string artist, string title) เป็นการเพิ่มเพลงดังกล่าวเข้าไปใน playlist ชื่อ name เพลงแต่ละเพลงอยู่ใน playlist ได้หลายอัน รับประกันว่าจะไม่มีการเรียกฟังก์ชันนี้โดยที่เพลงดังกล่าวยังไม่เคยถูกเรียก add_song มาก่อน

- vector<MP3> get_playlist_MP3(string name) เป็นฟังก์ชันที่จะถูกเรียกใช้เมื่อจะเล่นเพลงใน playlist ชื่อ name ฟังก์ชันนี้จะต้องคืนค่า vector ซึ่งประกอบด้วยข้อมูลส่วน mp3 ของเพลงแต่ละเพลงใน playlist ดังกล่าว รับประกันว่าจะไม่มีการเรียกฟังก์ชันนี้โดยที่ playlist name ยังไม่ได้สร้างขึ้นมาแน่นอน เมื่อเรียกฟังก์ชันนี้ เพลงแต่ละเพลงใน playlist นั้นถูกเล่น 1 ครั้ง
- vector<pair<string,string>> get_most_played(int k) เป็นฟังก์ชันที่จะคืนค่าข้อมูลส่วนศิลปินและชื่อเพลงจำนวน k เพลงที่ถูกเล่นมากที่สุด ให้ระวังว่าค่า k อาจจะมากกว่าเพลงที่มีอยู่ในระบบก็เป็นได้

ในโจทย์ข้อนี้ ให้ออกแบบคลาสโดยคำนึงว่าไฟล์เพลงแต่ละเพลงนั้นมีขนาดใหญ่มาก และเพลงแต่ละเพลงอาจจะอยู่ใน playlist ได้หลายอัน อันละหลายครั้ง พยายามเก็บข้อมูลให้ใช้เนื้อที่น้อยที่สุด นอกจากนี้ ให้คำนึงว่าเนื่องจากเป็นระบบออนไลน์ จำนวนเพลง และ จำนวน playlist นั้นมีจำนวนมาก ถ้าหากที่ไม่พอเขียน ให้เขียนไว้ด้านหลังของกระดาษแผนนี้ (หน้า 8) เท่านั้น

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>

Element Access	`operator[] (size_t n);`

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);`

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 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);
```