FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introduction to Data Structures
Year 2nd , First Semester , Midterm Examination 11 Oct 2018 8:30-11:30

ชื่อ-นามสกุล_____เลขประจำตัว_____ CR61___

หมายเหตุ

1. ข้อสอบมีทั้งหมด 11 ข้อในกระดาษคำถามคำตอบจำนวน 11 แผ่น 11 หน้า คะแนนเต็ม 100 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. (5 คะแนน) ตอบคำถามต่อไปนี้สั้น ๆ ว่าแต่ละปัญหาต้องมีที่เก็บข้อมูลประเภทใด

ตัวอย่าง: เก็บข้อมูลนิสิตว่าคนที่รหัสนี้มีชื่อนามสกุลอะไร

`map<string, pair<string, string>>` รหัส -> ชื่อ, นามสกุล

1.1 เก็บข้อมูลว่านิสิตที่รหัสนี้มีเกรดเฉลี่ยในแต่ละเทอมที่เรียนมาเท่าไหร่บ้าง

_____

1.2 เก็บข้อมูลรายการของหนังสือที่มีอยู่ในสต็อก โดยแต่ละเล่มระบุชื่อและราคา (หนังสืออาจมีชื่อซ้ำ แต่ราคาต่างกัน)

_____

1.3 เก็บว่าแต่ละวิชา(ระบุโดยรหัสวิชา) มีนิสิตคนใดลงทะเบียนบ้าง (ระบุโดยรหัสนิสิต)

_____

1.4 เก็บข้อมูลพิกัดสองมิติว่า มีอุณหภูมิในวันนี้กี่องศา

_____

1.5 เก็บว่านิสิตคนใด (ระบุโดยรหัส) เป็นสมาชิกสหกรณ์บ้าง เพื่อให้ตรวจสอบการเป็นสมาชิกได้เร็ว

_____

2. (5 คะแนน) จงอธิบายสั้น ๆ ได้ใจความว่าฟังก์ชันเหล่านี้ทำอะไร ได้ผลอะไร (ไม่ต้องอธิบายว่า แต่ละบรรทัดทำอะไร)

ฟังก์ชัน | ทำหน้าที่อะไร

2.1
```
float xxx(vector<float>& v) {
  float s = 0.0;
  for (auto& x:v) s+=x;
  return s / v.size();
}
```
_____

2.2
```
int xxx(vector<int> v) {
  sort(v.begin(), v.end());
  return v[v.size() – 2];
}
```
_____

2.3
```
void xxx(int[] d, int n) {
  priority_queue<int> q;
  for (int i=0; i<n; i++) q.push(d[i]);
  for (int i=0; i<n; i++)
     d[i] = q.top(); q.pop();
}
```
_____

2.4
```
void xxx(set<int> s) {
  return *(--(s.end()) - *(s.begin());
}
```
_____

2.5
```
float xxx(map<float, float>& a, float x) {
  for (auto& v:a)
    if (v.second == x)  return v.first;
}
```
_____

3. (5 คะแนน) จงวิเคราะห์อัตราการเติบโตของส่วนของโปรแกรมต่อไปนี้ ให้ตรงกับความเป็นจริงมากที่สุด โดยให้เขียนอยู่ในรูปสัญกรณ์เชิงเส้นกำกับในตัวแปร n

3.1
```
// v is a vector of n elements
int xxx(vector<int>& v, int e) {
  int k = 0;
  for (auto& x:v) if (x == e) k++;
  return k;
}
```
Θ(____)

3.2
```
// v is a vector of n elements, w is a map of m elements
int xxx(vector<int>& v, map<int, int>& w) {
  int k = 0;
  for (auto& x:v) k+=w[x];
  return k;
}
```
O(_____)

3.3
```
// v is a vector of n elements
int xxx(vector<int>& v){
    for (int i = 0; i < v.size() -1; i++)
       for (int j = i; j < v.size(); j++)
           if (v[i] == v[j]) return v[i];
}
```
Θ(____)

3.4
```
// v is a vector of n elements
void xxx(vector<int>& v) {
  priority_queue<int> q;
  for (auto& x:v) q.push(x);
  v.clear();
   while (!q.empty())  {v.push_back(q.top()); q.pop();}
}
```
O(_____)

3.5
```
// v is a vector of n elements
int xxx(vector<int>& v){
    for (int i = 0; i < v.size() -1; i++)
       for (int j = i; j < v.size(); j++)
           if (v[i] == v[j]) return v[i];
}
```
Θ(____)

4. (5 คะแนน) จงเติมส่วนของโปรแกรมให้มีเวลาทำงานดังที่กำหนด

4.1
```
// v is a vector of n elements
void xxx(vector<int>& v, int x) {

  v.insert(_____, x);
}
```
$\Theta(n)$

4.2
```
// s is a set of n elements
int xxx(set<int>& s, int x) {
  int k = 0;
  for (auto& e:s)  if (_____) k++;
  return k;
}
```
$\Theta(n \log n)$

4.3
```
// pq is a priority queue of n elements
void xxx(priority_queue<int>& pq, int x) {
  int k = 0;
  while (pq.size() > 0) {
    if (x > pq.top()) k++;

    _____
  }
  return k;
}
```
$\Theta(n \log n)$

4.4
```
// v is a vector of n elements
void xxx(vector<int>& v) {
  int k = 0;
  while (v.size() > 0) {
    if (k > v[0]) k++;

    v.erase(_____);
  }
  return k;
}
```
$\Theta(n)$

4.5
```
// v is an empty vector
void xxx(vector<int>& v, int n){
  int k = 0;
  for (int i = 0; i < n; i++) v.push_back(i);
  for (int i = 0; i < n; i++)
    k += *find(v.begin(), v.end(), _____);
  return k;
}
```
$\Theta(n^2)$

5. (20 คะแนน) จงเขียนฟังก์ชันที่ทำงานตามรายละเอียดที่เขียนใน comment

```
// ให้ m1 และ m2 เก็บความถี่ของคำ   merge จะนำข้อมูลใน m2 มาเพิ่มใน m1 เช่น
// ถ้าสร้าง m1 และ m2 ให้เป็น map ว่าง ๆ จากนั้นให้
// m1["one"] = 5; m1["two"] = 4; m2["two"] = 2; m2["ten"] = 1
// หลัง merge(m1,m2) จะได้ m1 มี key:value เป็น "one":5, "two":6, "ten":1
void merge(map<string,int> &m1, const map<string,int> &m2) {









}
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

```
//---------------------------------------------------------------------------
// คืน vector ของคู่ลำดับ i,j ที่ v[i][j] มีค่าเป็น 0
vector<pair<int,int>> get_zero_positions(const vector<vector<int>> & v) {









}
//---------------------------------------------------------------------------
// คืน map ที่เก็บว่า ใครถูกใคร follow จาก map ที่เก็บว่าใคร follow ใคร
// เช่น following เก็บ key:value เป็น { "A":{"a","b"}, "B":{"b","c"} }
// get_followers(following) จะได้ map ที่เก็บ { "a":{"A"}, "b":{"A","B"}, "c":{"B"} }
map<string, set<string>> get_followers(const map<string, set<string>> following) {









}
//---------------------------------------------------------------------------
// คืนเซตที่เก็บสมาชิกที่ปรากฏทั้งในเซต s1 และ s2 โดยห้ามใช้ฟังก์ชัน intersection, sort และ find
set<int> common(const set<int> &s1, const set<int> &s2) {









}
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

6. (10 คะแนน) เราสามารถสร้าง queue ด้วย priority queue และก็สามารถสร้าง stack ด้วย priority queue เช่นกัน จงเขียนเมท็อดต่าง ๆ ภายในคลาส my_queue และ my_stack ที่ภายในเก็บแค่ priority_queue และอนุญาตให้เพิ่มข้อมูลได้อีกหนึ่งตัวแบบ int, float, bool หรือ char

```
template <typename T>
class my_queue {
  protected:
    std::priority_queue<pair<int,T>> pq;




  public:
    ~my_queue() { }
    my_queue() { }
    my_queue(const my_queue<T>& q)  { ____________________________________________ }

    bool     empty() const          { ____________________________________________ }

    size_t   size() const           { ____________________________________________ }

    const T& top() const            { ____________________________________________ }

    void     pop()                  { ____________________________________________ }

    void     push(const T& element) {







    }
};
template <typename T>
class my_stack {
  protected:
    std::priority_queue<pair<int,T>> pq;




  public:
    ~my_stack() { }
    my_stack() { }
    my_stack(const my_queue<T>& q)  { ____________________________________________ }

    bool     empty() const          { ____________________________________________ }

    size_t   size() const           { ____________________________________________ }

    const T& top() const            { ____________________________________________ }

    void     pop()                  { ____________________________________________ }

    void     push(const T& element) {







    }
};
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

7. (10 คะแนน) จงเขียนเมท็อด **insert** และ **erase** ในคลาส **my_set** ข้างล่างนี้ให้สมบูรณ์ กำหนดให้สร้างเซตด้วยอาเรย์

```
template <typename T>
class my_set {
  protected:
    int * mData;  int   mCap;  int   mSize;
  public:
    my_set() {
      mCap = 1000; mData = new T[mCap](); mSize = 0;
    }
    my_set(const my_set<T>& s) {
      mData = new T[s.mCap]();
      mCap  = s.mCap;  mSize = s.mSize;
      for (size_t i = 0; i < s.mSize; i++) mData[i] = s[i];
    }
    ~my_set() { delete [] mData; }
    //-------------------------------------------
    typedef  T* iterator;
    iterator begin() { return mData; }
    iterator end()   { return mData + mSize; }
    size_t   size()  { return mSize; }
    bool     empty() { return mSize == 0;}
    iterator find(const T& val) {
      for (auto it = begin(); it != end(); ++it) {
        if (*it == val) return it;
        if (*it  > val) break;
      }
      return end();
    }
    //-------------------------------------------
    // เพิ่ม val เข้าไปในเซต ไม่ต้องห่วงเรื่องที่เก็บใน  mData เต็ม  คืนค่าจริง ถ้าเซตมีขนาดเพิ่มขึ้น ไม่เช่นนั้น คืนเท็จ
    bool insert(const T& val) {  // ห้ามใช้ฟังก์ชันใด ๆ ของ std *******
```

```
    }
    //-------------------------------------------
    // ลบข้อมูลออกจากเซตที่มีค่าเท่ากับ val เข้าไปในเซต  คืนค่าจริง ถ้าเซตมีขนาดลดลง ไม่เช่นนั้น คืนเท็จ
    bool erase(const T& val) {   // ห้ามใช้ฟังก์ชันใด ๆ ของ std *******
```

```
    }
};
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

8. (10 คะแนน) จงออกแบบโครงสร้างข้อมูลเพื่อเก็บข้อมูลพนักงานที่ทำตามข้อกำหนดเหล่านี้ ให้ n คือจำนวนพนักงานที่เก็บในฐานข้อมูล
   - เก็บชื่อ-นามสกุล และเลขประจำตัวประชาชนของพนักงานต่าง ๆ ได้
   - เพิ่มข้อมูลได้ใน O(log n)
   - ขอเลขประจำตัวประชาชนของพนักงานที่มีชื่อนามสกุลตามที่ระบุได้ใน O(log n), ถ้าไม่มีให้คืน string ว่าง
   - ขอรายการของเลขประจำตัวประชาชนของพนักงานที่มีนามสกุลตามที่ระบุในเวลา O(k log n) โดย k คือจำนวนคนที่มีนามสกุลที่ระบุ

นิสิตสามารถใช้ STL ใด ๆ ก็ได้โดยถือว่า ได้ include และ using namespace std ไว้เรียบร้อยแล้ว

```
class PersonDB{
private:




public:
  PersonDB() {



  }
  void addPerson(string& firstName, string& lastName, string& ID) {









  }
  string getIDofPerson(string& firstName, string& lastName) {









  }
  vector<string> getIDofPersonsWithLastName(string& lastName) {









  }
}
```

ชื่อ_____เลขประจำตัว_____เลขที่ใน CR61_____

9. (10 คะแนน) จงเพิ่มเมท็อด `eraseAllWithValue` ให้กับของ `CP::vector` เพื่อลบข้อมูลทุกตัวที่มีค่าเท่ากับ `x` โดยต้องทำงานในเวลา O(n) เมื่อ n คือจำนวนข้อมูลที่มีอยู่ก่อนเริ่มลบ นิสิตสามารถเรียกใช้บริการต่าง ๆ ที่มีอยู่แล้วใน `CP::vector` ได้

```
template <typename T>
class vector {
  public:
    typedef T* iterator;
  protected:
    T *mData;    size_t mCap;    size_t mSize;
    void rangeCheck(int n) {...}
    void expand(size_t capacity) {...}
    void ensureCapacity(size_t capacity) {...}
  public:
    ...
    void eraseAllWithValue(const T& x) {









    }
};
```

```
vector(const vector<T>& a) {...}
vector() {...}
vector(size_t cap) {...}
vector<T>& operator=(vector<T> other) {...}
~vector() {...}
bool     empty() const {...}
size_t   size() const {...}
size_t   capacity() const {...}
void     resize(size_t n) {...}
iterator begin() {...}
iterator end() {...}
T& at(int index) {...}
T& at(int index) const {...}
T& operator[](int index) {...}
T& operator[](int index) const {...}
void push_back(const T& element) {...}
void pop_back() {...}
iterator insert(iterator it,const T& element)
{...}
void erase(iterator it) {...}
void clear() {...}
```

10. (10 คะแนน) จงเพิ่มเมท็อด `insertAt` ให้กับ `CP::queue` เพื่อนำข้อมูลจากเวคเตอร์ `v` มาแทรกยังตำแหน่งที่ `k` จากหัวคิว (หัวคิวคือ `k = 0`) นิสิตสามารถเรียกใช้บริการต่างๆที่มีอยู่แล้วใน `CP::queue` และ `std::vector` ได้

```
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 std::vector<T>& v) {









        }
};
```

```
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_____

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);
```