FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introductions to Data Structure

YEAR II, Summer Semester, Final Examination, May 10, 2010, Time 09:00 – 12:00

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 7 ข้อในกระดาษคำถามคำตอบจำนวน 5 แผ่น 5 หน้า คะแนนเต็ม 70 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 (.....)

หมายเหตุ (เพิ่มเติม)

1. สำหรับข้อที่ให้ออกแบบ หรือ เขียนโปรแกรม คะแนนที่ได้จะแปรตามประสิทธิภาพในการทำงานของโปรแกรม
2. สำหรับข้อที่ให้วิเคราะห์เวลาการทำงาน คะแนนที่ได้จะแปรตามความใกล้เคียงความเป็นจริงของการวิเคราะห์
3. **นิสิตสามารถอ้างถึงและเรียกใช้คลาสต่าง ๆ ที่อยู่ในเอกสารประกอบการสอนได้โดยไม่จำเป็นต้องเขียนขึ้นมาใหม่**
4. สามารถเขียนคำตอบด้วยภาษา C++ หรือ Java ก็ได้ แต่ในกรณีที่เขียนด้วย Java นิสิตจะต้องแปลงโจทย์ให้เป็นภาษา Java ด้วยตัวเอง พร้อมทั้งเขียนโจทย์ที่แปลงลงในคำตอบด้วย

1. (10 คะแนน) จงเขียน constructor สำหรับคลาส `BSTree<T>` ที่รับข้อมูลเข้าเป็น `LinkedList<T>` ดังแสดงด้านล่างนี้ โดย constructor ดังกล่าวจะนำเอาข้อมูลทั้งหมดใน `LinkedList` ที่รับเข้ามาไปเพิ่มลงไปใน `BSTree`

```
template <class T>
class BSTree : public Set<T> {
public:
    BSTree(LinkedList<T> *list) {









    }
};
```

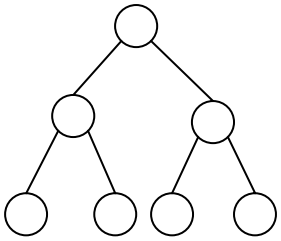
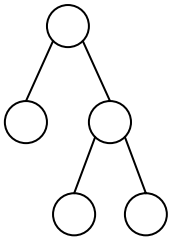
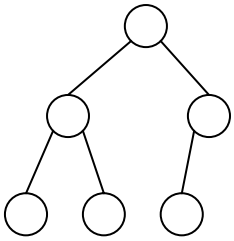
2. (10 คะแนน) จงเขียนฟังก์ชัน `void PrintDescSort()` สำหรับคลาส `BSTree<int>` ซึ่งเป็นโครงสร้างข้อมูลประเภท Binary Search Tree ที่เก็บข้อมูลประเภทเลขจำนวนเต็ม โดยฟังก์ชันดังกล่าวจะทำการพิมพ์ข้อมูลทั้งหมดใน `BSTree` ดังกล่าวโดยเรียงข้อมูลจาก**<u>มากไปน้อย</u>** (นิสิตสามารถเขียนฟังก์ชันเพิ่มเติมได้โดยอิสระ)

```
template <class T>
class BSTree : public Set<T> {
public:
    void PrintDescSort() {









    }
};
```

3. (10 คะแนน) จงเขียนฟังก์ชัน `void doReverse()` สำหรับคลาส `LinkedList<T>` โดยที่ฟังก์ชันดังกล่าวจะทำการพลิกกลับลำดับของข้อมูลใน `LinkedList` ดังกล่าวจากหัวเป็นท้าย ตัวอย่างเช่น ถ้าใน `LinkedList` มีข้อมูลเป็น `2,4,9,3,1` เมื่อเรียก `doReverse` แล้ว ข้อมูลใน `LinkedList` จะกลายเป็น `1,2,9,4,2` โดยมีข้อกำหนดเพิ่มเติมคือ ฟังก์ชันดังกล่าวจะต้องไม่มีการเปลี่ยนแปลงข้อมูลในตัวแปร element เด็ดขาด (ห้ามเขียนคำสั่ง `element = XXX` ใด ๆ ทั้งสิ้น) (หมายเหตุ: `LinkedList` ในที่นี่จะหมายถึง doubly linked list with header แบบไม่ใช่ circular)

```
template <class T>
class LinkedList : public List<T> {
public:
    void doReverse() {









    }
};
```

4. (10 คะแนน) จงเขียนฟังก์ชั่น `bool isFullBinary()` สำหรับคลาส `BinaryTree<int>` ซึ่งเป็นโครงสร้างข้อมูลที่เก็บปมของต้นไม้ทวิภาค เพื่อตรวจสอบว่าต้นไม้ย่อย (Subtree) ที่มีปมดังกล่าวเป็นรากนั้นมีลักษณะเป็นต้นไม้ทวิภาคแบบเต็มหรือไม่ โดยนิยามของต้นไม้ทวิภาคแบบเต็มคือทุก ๆ ปมภายใน (internal node) ของต้นไม้ดังกล่าวจะต้องมีลูกอยู่ 2 ปมเสมอ (ดูรูปด้านล่างประกอบ)

		
เป็นต้นไม้ทวิภาคแบบเต็ม	เป็นต้นไม้ทวิภาคแบบเต็ม	**ไม่เป็น**ต้นไม้ทวิภาคแบบเต็ม

```
template <class T>
class BinaryTree {
public:
    bool isFullBinary() {









    }
};
```

5. (5 คะแนน) จงเขียนฟังก์ชัน `int getMinAVLNode(int h)` ซึ่งคำนวณจำนวนปมน้อยที่สุดของต้นไม้เอวีแอลที่มีความสูงเป็น h (กำหนดให้ต้นไม้ที่ไม่มีข้อมูล มีความสูงเป็น -1 หรืออีกในหนึ่ง เมื่อ h มีค่าเป็น -1 จำนวนปมน้อยที่สุดจะเป็น 0)

```
int getMinAVLNode(int h) {









}
```

6. (10 คะแนน) สมมติให้มีฟังก์ชัน `int getMinAVLNode(int h)` ดังคำอธิบายในข้อ 5 ที่สามารถทำงานได้อย่างถูกต้องแล้ว จงเขียน ฟังก์ชัน `void genMinAVL(int h)` ซึ่งจะทำการสร้างต้นไม้ค้นหาแบบทวิภาค `BSTree<int>` **<u>ที่มีโครงสร้างตรงตามกฎของต้นไม้เอวีแอล</u>** โดยต้นไม้ดังกล่าวจะต้องมีข้อมูลประเภท int ที่มีค่าตั้งแต่ `start` ถึง `start + getMinAVLNode(h)` อยู่ การสร้างต้นไม้ดังกล่าวให้กระทำโดยการเรียกฟังก์ชัน `add(int x)` ของ `BSTree<int>` (คำแนะนำ: ใช้ฟังก์ชัน `getMinAVLNode` ให้เป็นประโยชน์)

```
void genMinAVL(int h,int start,BSTree<int>* avl) {









}
```

7. (15 คะแนน) จงออกแบบโครงสร้างข้อมูลสำหรับปัญหาต่อไปนี้ สมมติให้สมาคมกอล์ฟแห่งประเทศไทยต้องการที่จะจัดอันดับนักกอล์ฟสมัครเล่นในประเทศประจำปี 2009 โดยวิธีการจัดอันดับการแข่งขันเป็นดังนี้ ทุกครั้งที่นักกอล์ฟแต่ละคนไปเล่นกอล์ฟยังสนามต่าง ๆ นักกอล์ฟจะต้องบันทึกคะแนนที่ได้ (คะแนนที่ได้คือจำนวนครั้งในการตีให้ครบทุกหลุม ยิ่งตีน้อยครั้งแสดงว่ามีความสามารถมาก) อย่างไรก็ตาม นักกอล์ฟแต่ละคนนั้นจะเล่นกอล์ฟอยู่หลายสนาม และ สนามละหลาย ๆ ครั้ง ทางสมาคมจึงออกกฎว่า นักกอล์ฟจะต้องเก็บคะแนนเฉพาะ K ครั้งที่ดีที่สุด (จำนวนครั้งในการตีน้อยสุด) ของแต่ละสนามไว้ และระดับความสามารถของนักกอล์ฟคนนั้นก็คือ “ค่าเฉลี่ย” ของ “คะแนนเฉลี่ย” ของแต่ละสนาม โดยกำหนดให้มีสนามทั้งหมด N สนาม

จงเขียนคลาส ScoreCalc สำหรับคำนวณคะแนนของนักกอล์ฟแต่ละคน โดยคลาสดังกล่าวจะต้องมีฟังก์ชันดังต่อไปนี้

1) constructor ScoreCalc(int N, int K) เพื่อรับข้อมูล N และ K

2) ฟังก์ชัน void AddScore(int cIdx, int score) ที่ใช้เพิ่มข้อมูลคะแนนที่ได้ โดยตัวแปร cIdx เป็นหมายเลขของสนามที่เล่น มีค่าตั้งแต่ 0 ถึง N – 1 และตัวแปร score เป็นคะแนน (จำนวนครั้งในการตี) ที่นักกอล์ฟทำได้ในสนามดังกล่าว

3) ฟังก์ชัน float getProficiency() ซึ่งเป็นฟังก์ชันในการคำนวณระดับความสามารถของนักกอล์ฟ

นอกจากนี้ เราจะมีการกำหนดว่านักกอล์ฟแต่ละคนนั้นจะเล่นครบทั้ง N สนาม และ แต่ละสนามจะเล่นมากกว่า K ครั้งแน่นอน (เพื่อที่จะได้ไม่ต้องสนใจกรณีพิเศษต่าง ๆ) และค่า N และ K นั้นจะเป็นค่าจำนวนเต็มบวกที่มีค่าไม่เกิน 100

นิสิตสามารถนำเอาโครงสร้างข้อมูลอื่น ๆ มาใช้งานในคลาส ScoreCalc ได้อย่างอิสระ และสามารถเขียนฟังก์ชันเพิ่มเติมได้

```
class ScoreCalc {
public:
    ScoreCalc(int N,int K) {




    }

    void addScore(int cIdx,int score) {









    }

    float getProficiency() {









    }



private:



};
```