FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introductions to Data Structure
Year 2nd, Second Semester, Final Examination, February 23, 2010, Time 13:00 – 16:00

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 9 ข้อในกระดาษคำถามคำตอบ 7 หน้า คะแนนเต็ม 75 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. **ในข้อที่ต้องออกแบบโครงสร้างข้อมูล นิสิตไม่จำเป็นต้องตรวจสอบถึงกรณีที่มีการใส่ข้อมูลเข้าไปมากกว่าเนื้อที่ที่มีอยู่ (เสมือนว่าการจองพื้นที่นั้นจองมากพอเสมอ) หรือ กรณีที่เอาข้อมูลออกเมื่อไม่มีข้อมูลอยู่ในโครงสร้างข้อมูล**
5. **ให้เขียนคำตอบลงในเฉพาะพื้นที่ที่เว้นว่างไว้**
6. ให้นิสิตเขียนรหัสประจำตัวและเลขที่ใน CR58 ในทุกหน้าของกระดาษคำถามด้วย

---

1. (5 คะแนน) จงเขียนผลลัพธ์ของการเรียงข้อมูลต่อไปนี้ด้วย shell sort โดยใช้ h-sequence เป็น 4, 3, 1 ให้เขียนเฉพาะผลลัพธ์หลังจากการใช้ h แต่ละตัว

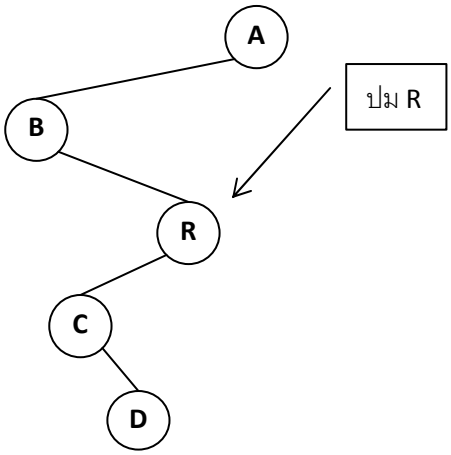
ข้อมูลเริ่มต้น	**7**	**3**	**6**	**10**	**8**	**1**	**2**	**9**
หลังใช้ h = 4								
หลังใช้ h = 3								
หลังใช้ h = 1								

2. (5 คะแนน) กำหนดให้มีตารางแฮช H ที่ใช้วิธีการแก้การชนแบบการตรวจกำลังสอง (Quadratic Probing) ที่มีขนาดเป็น 13 และสมมติให้แฮชฟังก์ชันของตัวแปรประเภท Integer ที่มีค่าเป็นเลขจำนวนเต็ม X คือ X % 13 (เศษของการหารด้วย 13) จงเขียนผลลัพธ์ของการเก็บข้อมูลในตารางแฮชหลังจากที่มีการเรียกใช้เมท็อด**ตามลำดับ**ดังต่อไปนี้ โดยให้เขียนเฉพาะผลลัพธ์ของการเก็บข้อมูลในตารางแฮช

1) H.add(new Integer(19));
2) H.add(new Integer(33));
3) H.add(new Integer(6));
4) H.add(new Integer(2));
5) H.remove(new Integer(6));
6) H.add(new Integer(35));
7) H.add(new Integer(22));
8) H.add(new Integer(45));

เขียนผลลัพธ์ลงในตารางต่อไปนี้

3. (5 คะแนน) สมมติให้มีต้นไม้ที่มีโครงสร้างดังรูปต่อไปนี้ จงวาดต้นไม้ผลลัพธ์ของการเรียกใช้ rotateLeftChild ที่ปม R

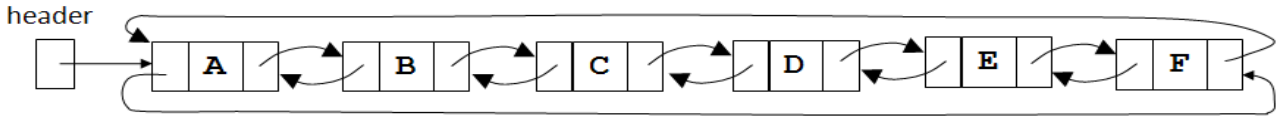
ต้นไม้ก่อนเรียก	ต้นไม้หลังเรียก (เขียนคำตอบ)
	

4. (10 คะแนน) จงเขียนเมท็อดการแบ่งส่วน (partition) สำหรับการเรียงลำดับแบบเร็ว (quick sort) โดยใช้วิธีเดียวกับที่ได้ศึกษาในชั้นเรียน สำหรับโครงสร้างข้อมูลแบบ LinkedList ซึ่งเป็นโครงสร้างข้อมูลประเภท circular doubly linked list with header

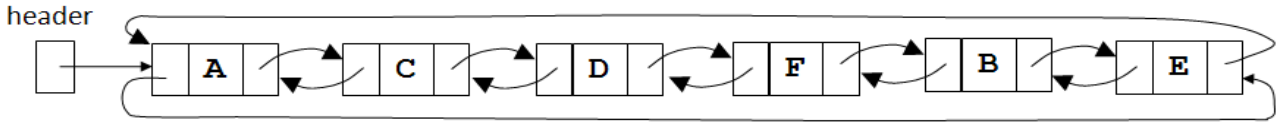
```
static int partition(LinkedList d, LinkedNode left, LinkedNode right) {




}
```

5. (10 คะแนน) กำหนดให้มีคลาส PhotoList ซึ่งเก็บข้อมูลรายการของรูปถ่ายหลาย ๆ รูปแบบมีลำดับ โดยใช้หลักการเช่นเดียวกับ circular doubly linked list with header ซึ่งมีโครงสร้างดังส่วนของโปรแกรมข้างล่างนี้ จงเขียนเมท็อด reorder สำหรับคลาสดังกล่าวเพื่อทำการเรียงรูปตามที่เราต้องการ โดยเมท็อด reorder นั้นจะรับจำนวนเต็ม int x และ array ของจำนวนเต็ม int [] pics และจะทำการจัดเรียงรูปต่าง ๆ โดยมีหลักการดังนี้ เมท็อด reorder จะเลื่อนรูปทุกรูปที่อยู่ที่ตำแหน่งต่าง ๆ ตามที่ระบุใน int [] pics ไปแทรกไว้ในตำแหน่ง x โดยที่รูปที่เคยอยู่ ณ ตำแหน่ง pics[0] จะย้ายไปอยู่ที่ตำแหน่ง x, รูปที่เคยอยู่ ณ ตำแหน่ง pics[1] จะย้ายไปอยู่ที่ตำแหน่ง x + 1,... กำหนดให้รูปแรกในรายการนั้นมีตำแหน่งเป็น 0 และรูปถัด ๆ ไปมีตำแหน่งเป็น 1, 2, … และกำหนดให้ข้อมูลในตัวแปร int [] pics นั้นเรียงจากน้อยไปมากอยู่แล้ว และค่า x นั้นจะน้อยกว่า pics[0] หรือ มากกว่า pics[pics.length – 1] เสมอ ตัวอย่างในรูปต่อไปนี้แสดงถึงผลลัพธ์ของการเรียกใช้เมท็อด reorder(x,pics) เมื่อ x มีค่าเป็น 1 และ pics มีค่าเป็น (2,3,5)

<u>ก่อนเรียก reorder(x,pics)</u>



<u>หลังจากเรียก</u>




```
public class PhotoList {
  private static class PhotoNode {
    Object element;
    LinkedNode prev, next;
    LinkedNode(Object e, LinkedNode p, LinkedNode n) {
      this.element = e; this.prev = p;  this.next = n;
    }
  }

  private LinkedNode header;


  public void reorder(int x,int [] pics) {









  }
}
```

6. (10 คะแนน) จงปรับปรุงโครงสร้างข้อมูล LinearProbingHashSet โดยให้เขียนเมท็อด LinkedList toList() ซึ่งจะ return โครงสร้างข้อมูลแบบ circular doubly linked list with header ที่มีข้อมูลเป็นสมาชิกทุกตัวในตารางแฮช โดยที่ toList() นั้นจะใช้เวลาการทำงานเป็น O(N) เมื่อ N เป็นจำนวนข้อมูลในตารางแฮช **(ไม่ใช่**ขนาดของตารางแฮช) และ กำหนดให้โครงสร้างข้อมูลดังกล่าวไม่มีการลบข้อมูลออก (ไม่มีเมท็อด remove และไม่จำเป็นต้องใช้) นิสิตสามารถเพิ่ม field สมาชิก และ/หรือ แก้ไขเมท็อด add(Object e) ตามที่เห็นสมควร (หมายเหตุ: เมท็อด oldAdd(Object e) นั้นเป็นเมท็อด add ตามปรกติของ LinearProbingHashSet ที่ให้ไว้เพื่ออ้างอิง นิสิตสามารถเรียกใช้เมท็อดอื่น ๆ ของ LinearProbingHashSet ได้ตามปรกติ และให้คิดว่าขนาดของตาราง hash นั้นใหญ่มากพอที่จะเก็บข้อมูลได้เสมอ (ไม่จำเป็นต้อง rehash))

```
public class LinearProbingHashSet implements Set {
  private static final Object DELETED = new Object();
  private Object[] table;
  private int size = 0;
  private int numNonNulls = 0;
  // นิสิตสามารถเพิ่มเติม field ได้ตามที่เห็นสมควร และ เมท็อดอื่น ๆ ที่มีสามารถเรียกใช้ได้ตามปรกติ




  public void oldAdd(Object e) {
    int i = indexOf(e);
    if (table[i] == null) {
      table[i] = e;
      ++size; ++numNonNulls;
      if (numNonNulls > table.length/2) rehash();
    }
  }

  public void add(Object e) {









  }

  public LinkedList toList() {









  }
}
```

7. (10 คะแนน) ประสิทธิภาพของโครงสร้างข้อมูลประเภท BinarySearchTree (BST) นั้นขึ้นอยู่กับความสูงของต้นไม้ เราต้องการให้ต้นไม้นั้นมีความสูงน้อยที่สุดเท่าที่เป็นไปได้ สมมติว่า เราต้องการใส่ข้อมูลจำนวนหลาย ๆ ตัวลงไปใน BST ที่เริ่มต้นที่ยังไม่มีข้อมูลเลย การเลือกข้อมูลไปเพิ่มใส่ใน BST ย่อมมีผลต่อความสูงของต้นไม้

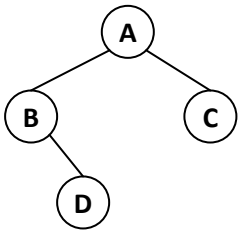
   1) กำหนดให้ข้อมูลที่ต้องการใส่ลงไปใน BST มีดังต่อไปนี้ 2, 4, 6, 8, 10, 12, 14 จงเขียนลำดับของการใส่ข้อมูลลงไปใน BST ที่ทำให้ความสูงของ BST นั้นน้อยที่สุด

      ……
      ……
      ……
      ……

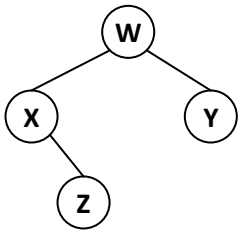
   2) จงเขียน Constructor ของ BSTree ที่รับข้อมูลเป็น array ของ Object ที่เรียงจากน้อยไปมากมาเรียบร้อยแล้ว โดย constructor ดังกล่าวจะต้องสร้าง BST ที่ประกอบด้วยข้อมูลตาม array ที่ได้รับมา ให้เตี้ยที่สุดเท่าที่เป็นไปได้

```
public class BSTree extends BinaryTree {

  //method และ field อื่น ๆ ที่มีของ BSTree และ BinaryTree สามารถเรียกใช้ได้ตามปรกติ

  public BSTree(Object [] sorted) {









  }
}
```

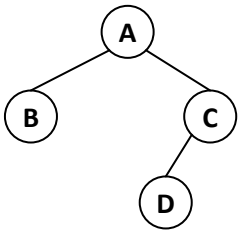
8. จงเขียนเมท็อด boolean isSameStructure(BinaryTree that) สำหรับโครงสร้างข้อมูลประเภท BinaryTree เพื่อตรวจสอบว่า BinaryTree สองต้นนั้นมี**รูปร่าง**เหมือนกันหรือไม่ โดยไม่คำนึงถึงข้อมูลภายใน ดังตัวอย่างในรูปต้นไม้สามต้นต่อไปนี้ สองต้นแรกจะมีรูปร่างเหมือนกัน ในขณะที่ต้นที่สามจะไม่เหมือนกับสองต้นแรก (คำแนะนำ: ควรใช้การเขียนโปรแกรมแบบเวียนบังเกิด)



ต้นไม้ต้นที่ 1



ต้นไม้ต้นที่ 2



ต้นไม้ต้นที่ 3

```
public class BinaryTree {
  //method และ field อื่น ๆ ที่มีของ BinaryTree สามารถเรียกใช้ได้ตามปรกติ
  public boolean isSameStructure(BinaryTree that) {









  }
}
```

9. (10 คะแนน) จงออกแบบโครงสร้างข้อมูลสำหรับเว็บไซต์ซึ่งมีรายละเอียดดังต่อไปนี้ เว็บไซต์ชื่อ tweeter เป็นเว็บไซต์ที่มีการทำงานทำนองเดียวกับเว็บไซต์ www.twitter.com โดยเว็บไซต์ tweeter จะมีระบบสมาชิก โดยมีสมาชิกทั้งหมด N คน สมาชิกแต่ละคนจะได้รับ ID ซึ่งแทนด้วยตัวเลข 0, 1, 2, …, N – 1 สมาชิกแต่ละคนสามารถเขียนข้อความลงไปใน account ของตัวเองได้ โดยข้อความแต่ละข้อความจะทราบถึงลำดับที่แต่ละข้อความได้ถูกบันทึกเข้าสู่เว็บไซต์ นอกจากนี้ สมาชิกแต่ละคนจะสามารถเรียกดูข้อความของสมาชิกอื่น ๆ ได้ โดยการเรียกดูข้อความของสมาชิกคนอื่น ๆ นั้น สมาชิกคนที่เรียกดูนั้นจะต้องระบุรายการของสมาชิกอื่น ๆ ที่ต้องการดู (เรียกว่ารายการ “**เพื่อนสมาชิก**”) และระบบจะทำการสร้างรายการของข้อความของสมาชิกตามที่ได้ระบุไว้ในรายการ**เพื่อนสมาชิก** ซึ่งรายการดังกล่าวนั้นจะประกอบด้วยข้อความทั้งหมดของสมาชิกทุกคนในรายการ**เพื่อนสมาชิก** โดยข้อความจะเรียงตามเวลาที่สมาชิกแต่ละคนได้เขียนข้อความลงไปในระบบ

จงออกแบบโครงสร้างข้อมูล TweetEngine ซึ่งต้องมีเมท็อดที่จำเป็น 3 เมท็อด ได้แก่ 1) TweetEngine(int N) เป็น constructor เพื่อสร้างโครงสร้างข้อมูลสำหรับระบบที่มีสมาชิก N คน 2) เมท็อด void doPost(int userID, String message) ซึ่งจะเป็นการเขียนข้อความของสมาชิก หมายเลข userID ด้วยข้อความ message และ 3) เมท็อด LinkedList read(int [] friends) ซึ่งจะเป็นการอ่านข้อความของ**เพื่อนสมาชิก**โดย จะ return กลับมาเป็น array ของ String ที่ประกอบด้วย message ของสมาชิกที่มีหมายเลขตามที่ระบุใน array friends โดย message ใน array นั้นจะต้องเรียงตามเวลาที่ได้เรียก void doPost(int userID, String message) ไว้ก่อนหน้านี้

ตัวอย่างต่อไปนี้แสดงถึงผลการทำงานของโครงสร้างข้อมูล TweetEngine engine ซึ่งมีสมาชิกจำนวน 3 คน

```
engine = new TweetEngine(3);
engine.doPost(0,“hi! a-0”);
engine.doPost(1,“ hi! b-0”);
engine.doPost(0,“ hi! a-1”);
engine.doPost(2,“ hi! c-0”);
engine.doPost(0,“hi! a-2”);
engine.doPost(2,“hi! c-1”);
int [] friend = {0, 2};
LinkedList msg = engine.read(friend);
```

บรรทัดสุดท้ายจะได้ผลลัพธ์เป็น (“hi! a-0”, “hi! a-1”, “hi! c-0”, “hi! a-2”, “hi! c-1”)

1) จงอธิบายแนวคิดโดยสังเขปของโครงสร้างข้อมูล TweetEngine

……
……
……

2) จงเขียนคลาส TweetEngine ในพื้นที่ว่างหลังจากนี้ (ถ้าพื้นที่ไม่พอให้เขียนต่อหน้าหลังของกระดาษแผ่นนี้ได้)