FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introductions to Data Structure

YEAR II, Second Semester, Midterm Examination, December 22, 2009, Time 13:00 – 16:00

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 9 ข้อในกระดาษคำถามคำตอบจำนวน 7 แผ่น 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) จงเรียงอัตราการเจริญเติบโตของฟังก์ชันต่อไปนี้จากน้อยไปหามาก โดยให้เขียนสัญลักษณ์ (a), (b), (c) แทนแต่ละฟังก์ชันตามตารางต่อไปนี้ (ในกรณีที่ฟังก์ชันมีอัตราการเจริญเติบโตเท่ากัน สามารถเรียงอันใดขึ้นก่อนก็ได้)

   (a) $4N + 5$ (b) $N/\sqrt{N}$ (c) $N!$ (d) $\log N^2$ (e) $\max(2N, N^2)$ (f) $\log N$ (g) $\log N * \log N$

   ตอบ:.....

2. (5 คะแนน) จงหาค่าของนิพจน์เติมหลัง 4 2 – 5 ↑ 3 5 + 4 8 – * + โดยใช้กองซ้อน (Stack) กำหนดให้เครื่องหมาย ↑ แทนการยกกำลัง ให้แสดงเฉพาะผลลัพธ์สุดท้ายที่ได้ และ ให้แสดงสถานะของกองซ้อนในขณะที่มีข้อมูลในกองซ้อนมากที่สุด

   .....
   .....
   .....
   .....
   .....
   .....

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

3. (5 คะแนน) ปัญหา MCP เป็นเกม puzzle เกี่ยวกับสายอักขระ (String) โดยมีกติกาดังนี้ ผู้เล่นจะเริ่มต้นด้วยสายอักขระ P และผู้เล่นจะสามารถสร้างอักขระใหม่จาก อักขระที่มีอยู่ ตามกฎ 4 ข้อ ดังต่อไปนี้

   1) ถ้าสายอักขระที่มีอยู่ลงท้ายด้วย P เราสามารถสร้างสายอักขระใหม่โดยเติม C ต่อท้ายไปได้ เช่น P →PC
   2) เราสามารถสร้างสายอักขระใหม่ โดยการทำสำเนาซ้ำต่อท้ายอีกรอบ เช่น P → PP หรือ CPC → CPCCPC
   3) ถ้าสายอักขระที่มีอยู่ มี PPP เป็นส่วนประกอบ เราสามารถเปลี่ยน PPP เป็น C ได้ เช่น CPPPC → CCC
   4) ถ้าสายอักขระมี CC เป็นส่วนประกอบ เราสามารถลบ CC ทิ้งไปได้ เช่น PCCP → PP

   ปัญหา MCP ต้องการทราบว่า เริ่มจากตัว P เราจะสามารถสร้างสายอักขระ CP ขึ้นมาได้จากจากกฎดังกล่าวหรือไม่ โปรแกรมต่อไปนี้เป็นโปรแกรมเพื่อแก้ปัญหาดังกล่าว จงเลือกโครงสร้างข้อมูลที่เหมาะสมเพื่อใช้แก้ปัญหาดังกล่าว โดยให้ตอบว่า โครงสร้างข้อมูล XXX และ YYY ในโปรแกรมดังกล่าวควรจะเป็นโครงสร้างข้อมูลใด

```
public static void bfsMCP() {
  XXX data1 = new XXX(...);
  YYY data2 = new YYY(...);
  String st = "P";
  data1.เพิ่ม(st); data2.เพิ่ม(st);
  while( !data1.ว่างหรือไม่() ) {
    st = (String) data1.เอาออก();
    if (st.equals("CP")) break;
    if (st.charAt(st.length() – 1) == 'P')
      if (!data2.มีหรือไม่(st + "C")) { data1.เพิ่ม(st + "C"); data2.เพิ่ม(st + "C");}
    if (!data2.มีหรือไม่(st + st)) { data1.เพิ่ม(st + st); data2.เพิ่ม(st + st);}
    for(int i = 0; i <= st.length()-3; i++)
      if (st.substring(i,i+3).equals(“PPP”)) {
        String v1 = st.substring(0,i)+”C”+st.substring(i+3,st.length());
        if (!data2.มีหรือไม่(v1)) { data1.เพิ่ม(v1); data2.เพิ่ม(v1);}
      }
    for(int i = 0; i <= st.length()-2; i++)
      if (st.substring(i,i+2).equals("CC")) {
        String v2 = st.substring(0,i) + st.substring(i+2,st.length());
        if (!data2.มีหรือไม่(v2)) { data1.เพิ่ม(v2); data2.เพิ่ม(v2);}
      }
  }
  if (st.equals("CP")) System.out.println("YES!!!");
}
```

ตอบ: โครงสร้างข้อมูล XXX คือ .....

โครงสร้างข้อมูล YYY คือ .....

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

4. (10 คะแนน) จงเขียนเมท็อด `remove(int k)` สำหรับโครงสร้างข้อมูลฮีปทวิภาค เพื่อลบข้อมูลในอาเรย์ช่องที่ k ออก โดยที่ยังคงรักษาความเป็นฮีปทวิภาคไว้ได้ เมท็อดดังกล่าวควรจะใช้เวลาในการทำงานเป็น $O(\lg N)$ เมื่อ N เป็นจำนวนข้อมูลในฮีป

```
public class BinaryHeap implements PriorityQueue {
  private Object[] elementData;
  private int size;
  // ไม่ได้แสดง method ต่าง ๆ ของ BinaryHeap  แต่สามารถเรียกใช้ได้ตามปรกติ
  public Object remove(int k) {









  }
}
```

5. (10 คะแนน) จงออกแบบโครงสร้างข้อมูล QueueByBinaryHeap ซึ่งเป็นโครงสร้างข้อมูลประเภทแถวคอย โดยที่โครงสร้างข้อมูลนี้จะใช้โครงสร้างข้อมูลแบบ BinaryHeap ที่เก็บข้อมูล ให้นิสิตเขียนโปรแกรมเพิ่มเติมในเมท็อด public void enqueue(Object e) และเมท็อด public Object dequeue() จากโปรแกรมภาษาจาวาด้านล่างนี้ ห้ามนิสิตทำการเพิ่มเติม field สมาชิก ใด ๆ ลงใน class ดังกล่าว (**หมายเหตุ: ไม่จำเป็นต้องตรวจสอบกรณีที่มีการใส่ข้อมูลเมื่อแถวคอยนี้เต็ม หรือการเอาข้อมูลออกเมื่อแถวคอยนี้ไม่มีข้อมูล คำแนะนำ: นิสิตควรจะใช้คลาสย่อย Element และตัวแปร useMe ในการทำงาน**)

```
public class QueueByPriorityQueue {
   BinaryHeap bh;
   int useMe;              // <--- คำแนะนำ: ควรใช้ตัวแปรตัวนี้ให้เป็นประโยชน์
   class Element implements Comparable {









   }
   public void enqueue(Object e) {




   }
   public Object dequeue() {




   }
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

6. (10 คะแนน) โครงสร้างข้อมูลแบบกองซ้อนตามที่ได้เรียนมานั้น เมท็อด void push(Object e) และ Object pop() นั้นจะกระทำกับข้อมูลทางด้าน “บน” ของกองซ้อน ให้นิสิตออกแบบโครงสร้างข้อมูลที่ชื่อว่า TwoWayStack ซึ่งทำงานเหมือนกองซ้อนปรกติทุกอย่าง แต่มีเมท็อดเพิ่มเติม คือ void pushBottom(Object e) และ Object popBottom() ซึ่งจะใส่ข้อมูลเข้า และ นำข้อมูลออกจากด้านล่างของกองซ้อน เมท็อดใหม่และเก่าทั้งหมดควรจะมีประสิทธิภาพเชิงเวลาเป็น $\Theta(1)$ จงเติมส่วนของโปรแกรมลงในช่องว่างข้างล่างนี้ (**หมายเหตุ: ไม่จำเป็นต้องตรวจสอบกรณีที่มีการใส่ข้อมูลเมื่อกองซ้อนนี้เต็ม หรือการเอาข้อมูลออกเมื่อกองซ้อนนี้ไม่มีข้อมูล คำแนะนำ: นิสิตอาจจะต้องประกาศ field สมาชิกเพิ่มเติม และอาจจะต้องแก้ไข constructor**)

```
public class TwoWayStack {
   private Object[] elementData;
   private int size;


   public TwoWayStack(int cap) {
     elementData = new Object[cap];



   }
   public Object pop() {









   }
   public void push(Object e) {









   }
   public Object popBottom() {









   }
   public void pushBottom(Object e) {









   }
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

7. (10 คะแนน) ฮีปแบบทวิภาคนั้นมีโครงสร้างเป็นต้นไม้ทวิภาค (binary tree) ให้นิสิตทำการดัดแปลงฮีปแบบทวิภาค โดยเปลี่ยนจากการใช้ต้นไม้ทวิภาค เป็นต้นไม้ไตรภาค (3-ary tree) ซึ่งต้นไม้ดังกล่าวจะมีลูกจำนวน 3 ลูก แทนที่จะมีแค่ 2 ลูก โดยที่หลักการอื่น ๆ ยังคงเหมือนเดิม

   1) จงอธิบายวิธีการระบุตำแหน่งของลูกทั้ง 3 ใน อาเรย์เมื่อกำหนดให้ตำแหน่งของปมพ่อนั้นอยู่ที่ตำแหน่ง k

      .....
      .....
      .....
      .....

   2) จงอธิบายวิธีการระบุตำแหน่งของพ่อ เมื่อกำหนดให้ตำแหน่งของปมลูกนั้นอยู่ที่ตำแหน่ง k

      .....
      .....

   3) จงเขียนเมท็อด fixDown(int k) สำหรับโครงสร้างข้อมูลนี้ **พร้อมทั้งวิเคราะห์ประสิทธิภาพเชิงเวลาของเมท็อดดังกล่าว**

```
public class BinaryHeap implements PriorityQueue {
  private Object[] elementData;
  private int size;

  // ไม่ได้แสดง method ต่าง ๆ ของ BinaryHeap  แต่สามารถเรียกใช้ได้ตามปรกติ

  private void fixDown(int k) {









  }
}
```

การวิเคราะห์ประสิทธิภาพเชิงเวลาของเมท็อด `fixDown` ที่เขียน

.....
.....
.....
.....
.....
.....

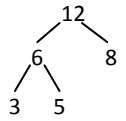
ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

8. (10 คะแนน) กำหนดให้ฮีปทวิภาค A มีข้อมูล n ตัวประกอบด้วยข้อมูล $(a_1, a_2, a_3, \dots, a_n)$ โดยที่ $a_1 \geq a_2 \geq a_3 \geq \cdots \geq a_n$ และให้ฮีปทวิภาค B มีข้อมูล m ตัวประกอบด้วยข้อมูล $(b_1, b_2, b_3, \dots, b_m)$ โดยที่ $b_1 \geq b_2 \geq b_3 \geq \cdots \geq b_m$ (ขอเน้นว่า subscript ที่เขียนใน $(a_1, a_2, a_3, \dots, a_n)$ และ $(b_1, b_2, b_3, \dots, b_m)$ ไม่จำเป็นต้องเป็น index ของอาเรย์ที่แทนฮีป)

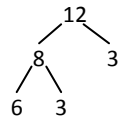
จงเขียนเมท็อด int compareTo(Object obj) เพื่อทำให้คลาส BinaryHeap เป็นแบบ Comparable

โดยมีนิยามการเปรียบเทียบฮีปสองต้นดังนี้

- ฮีป A จะ "มากกว่า" ฮีป B เมื่อ $a_1 > b_1$ ในกรณีที่ $a_1 = b_1$ ฮีป A จะ "มากกว่า" ฮีป B ก็ต่อเมื่อ ฮีป A' ที่เก็บ $(a_2, a_3, \dots, a_n)$ มากกว่า ฮีป B' ตัว ที่เก็บ $(b_2, b_3, \dots, b_m)$ ตัวอย่างเช่น ฮีปสองตัวที่แสดงทางขวานี้ ฮีปตัวบน "มากกว่า" ฮีปตัวล่าง



- ฮีปที่มีข้อมูลจะ "มากกว่า" ฮีปว่าง (ไม่มีสมาชิกเลย)
- ฮีปสองต้นจะ "เท่ากัน" ก็ต่อเมื่อมีข้อมูลในฮีปเหมือนกันทุกตัว



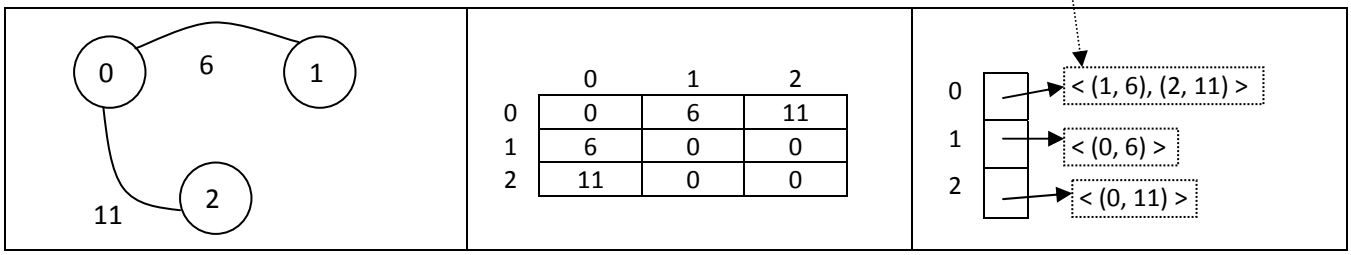
หมายเหตุ : การทำงานของ compareTo ต้องไม่เปลี่ยนข้อมูลภายในฮีป นั่นคือหลังการเรียกใช้ compareTo ข้อมูลภายในฮีปทั้งสองจะต้องเหมือนเดิม

```
public class BinaryHeap implements PriorityQueue, Comparable {
  private Object[] elementData;
  private int size;

  // ไม่ได้แสดง method ต่าง ๆ ของ BinaryHeap แต่สามารถเรียกใช้ได้ตามปรกติ

  public int compareTo(Object obj) {
     BinaryHeap that = (BinaryHeap) obj;









  }
}
```

9. (10 คะแนน) กราฟถ่วงน้ำหนักแบบไม่มีทิศทาง (undirected weighted graph) เป็นแบบจำลองทางคณิตศาสตร์ที่ประกอบด้วยเซตของปมและเซตของเส้นเชื่อม โดยที่เส้นเชื่อมจะเป็นคู่อันดับของปม และ แต่ละเส้นเชื่อมจะมีค่าน้ำหนักกำกับอยู่ กำหนดให้ เมทริกซ์ประชิด (Adjacency Matrix) $A = [a_{ij}]$ ของกราฟดังกล่าวที่ประกอบด้วยปมจำนวน N ปมเป็นเมทริกซ์ขนาด N × N โดยที่ a_{ij} จะมีค่าเท่ากับน้ำหนักของเส้นเชื่อมที่เชื่อมจากปม i ไปยังปม j และจะมีค่าเท่ากับ a_{ji} กำหนดให้รายการประชิด (Adjacency List) ของกราฟดังกล่าวนั้นประกอบด้วยรายการ (List) จำนวน N รายการ สำหรับแต่ละปม โดยที่รายการสำหรับปม p จะเก็บคู่อันดับที่ระบุถึง

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

ปมทั้งหมดที่เชื่อมกับปม p พร้อมด้วยค่าถ่วงน้ำหนักของเส้นเชื่อมที่เชื่อมปมนั้น ตัวอย่างต่อไปนี้แสดงกราฟ พร้อมด้วย เมทริกซ์ประชิด และ รายการประชิด ของกราฟดังกล่าว



กราฟ เมทริกซ์ประชิด รายการประชิด

จงเขียนเมท็อดที่ทำการแปลงเมทริกซ์ประชิดเป็นรายการประชิด และเมท็อดที่ทำการแปลงรายการประชิดเป็นเมทริกซ์ประชิด

```
public class GraphUtil {
   static class ListItem {
      int toNode, weight;
      public ListItem(int toNode, int weight) {
         this.toNode = toNode; this.weight = weight;
      }
   }
   public static ArrayList [] MatrixToList(int [][] adjMat) {
      int size = adjMat.length;
      ArrayList [] list = new ArraList[size];









      return list;
   }
   public static int [][] ListToMatrix(ArrayList [] adjList) {
      int size = adjList.length;
      int [][] adjMat = new int[size][size];









      return adjMat;
   }
}
```