FACULTY OF ENGINEERING
CHULALONGKORN UNIVERSITY
2110211 Introductions to Data Structure

YEAR II, Second Semester, Final Examination, March 1, 2011, Time 13:00 – 16:00

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

หมายเหตุ

1. ข้อสอบมีทั้งหมด 8 ข้อในกระดาษคำถามคำตอบจำนวน 7 แผ่น 7 หน้า คะแนนเต็ม 82 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. (12 คะแนน) จงตอบคำถามต่อไปนี้ ในช่องว่างที่เว้นไว้

   1.1. ให้ q เป็นตัวแปรอ้างอิงอ็อบเจกต์แบบ LinkedNode (มี element และ next เป็นตัวแปรภายใน) ของ list แบบ singly linked list ไม่มี header ถ้าต้องการลบปมหลังปมที่ q อ้างอิงอยู่ ต้องเขียนคำสั่งดังนี้ ……

   1.2. จริงหรือไม่ที่เราสามารถใช้ circular singly linked list with header มาสร้าง queue ได้ทันที โดยที่ enqueue และ dequeue ใช้เวลา O(1) ……

   1.3. จริงหรือไม่ที่เราสามารถใช้ singly linked list (ที่ไม่ circular และไม่มี header) มาสร้าง stack ได้ทันที โดยที่ push และ pop ใช้เวลา O(1) ……

   1.4. ให้ T คือ binary search tree ว่าง ๆ ต้นหนึ่ง ถ้าเพิ่มข้อมูล 3,4,1,5,7,9,6,0 เข้าไปใน T ตามลำดับจากซ้ายไปขวาที่เขียน T จะมีความสูงเป็น ……

   1.5. ให้ T คือ AVL tree ว่าง ๆ ต้นหนึ่ง ถ้าเพิ่มข้อมูล 3,4,1,5,7,9,6,0 เข้าไปใน T ตามลำดับจากซ้ายไปขวาที่เขียน T จะมีความสูงเป็น ……

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

1.6. binary search tree ที่เก็บข้อมูลจำนวน 700 ตัว มีความสูงได้น้อยสุดเท่ากับ .....

1.7. จริงหรือไม่ว่า ถ้ารากของ binary search tree ไม่มีลูกขวา (แต่มีลูกซ้าย) ค่ามากสุดของต้นนี้ต้องอยู่ที่ราก .....

1.8. ต้นไม้ AVL ที่มีความสูง 6 มีจำนวนปมน้อยสุด .....ปม

1.9. ต้นไม้ AVL ที่มีความสูง 6 มีจำนวนปมมากสุด .....ปม

1.10. ถ้าเก็บประวัตินิสิตวิศวคอมฯ จุฬาฯ (ที่มีประมาณ 500 คน) โดยใช้ตารางแฮชขนาด 1000 ช่อง และ h(x) = x % 1000 โดยที่ x คือรหัสนิสิต ย่อมไม่ดีแน่ เพราะ.....

1.11. ให้ H คือตารางแฮชแบบ open addressing ขนาด M ช่อง ถ้า H มี load factor เป็น L การหาว่าข้อมูลใดที่เก็บใน H มีค่ามากสุดจะใช้เวลาการทำงานเป็น ...Θ(.....)...

1.12. ถูกหรือผิดที่สมชายแนะนำว่า หากใช้ตารางแฮชแบบ separate chaining ก็ต้องระวังเรื่องอย่าให้ load factor เกิน 0.5 .....

2. (10 คะแนน) สำหรับคลาส LinkedList (circular doubly linked list with header) จงเขียนเมท็อด public void swap(int a,int b) ซึ่งจะทำการสลับปมที่เก็บข้อมูลที่ตำแหน่ง a กับ ตำแหน่ง b ของ list ที่เรียก โดยที่ $0 \leq a \leq b < size$ การเขียนเมท็อด swap นี้ห้ามเรียกใช้สมาชิก element ของคลาส LinkedNode ให้ใช้เฉพาะสมาชิก prev และ next เท่านั้น

```
public class LinkedList implements List {
  private static class LinkedNode {
    Object element;
    LinkedNode prev, next;
    LinkedNode(Object e, LinkedNode p, LinkedNode n) {
      this.element = e; this.prev = p;  this.next = n;
    }
  }
  private LinkedNode header;
  //ไม่ได้แสดงเมท็อดหรือสมาชิกอื่น ๆ ของ LinkedList แต่สามารถเรียกใช้ได้ตามปรกติ
  public void swap(int a, int b) {









  }
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

3. (10 คะแนน) สำหรับคลาส LinkedList (circular doubly linked list with header) จงเขียนเมท็อด public void append(LinkedList that) ซึ่งจะนำข้อมูลใน that มาต่อท้ายข้อมูลของ list นี้ เช่น ให้ x เป็น list เก็บ <a, b, c, d> และให้ y เป็น list เก็บ <p, q, r> ถ้าเราเรียก x.append(y) x จะกลายเป็น <a, b, c, d, p, q, r> และ y จะกลายเป็น list ที่ไม่มีข้อมูล จงเขียนเมท็อด append ที่ใช้เวลาในการทำงานเป็น $\Theta(1)$

```
public class LinkedList implements List {
  //ไม่ได้แสดงเมท็อดหรือสมาชิกอื่น ๆ ของ LinkedList แต่สามารถเรียกใช้ได้ตามปรกติ
  public void append(LinkedList that) {









  }
}
```

4. (10 คะแนน) สำหรับคลาส AVLTree นั้น เรารักษากฎความสูงของต้นไม้ได้โดยเมท็อด rebalance(Node r) ซึ่งจะทำการแก้ไขต้นไม้ย่อยที่มี r เป็นรากตามแต่กรณีของการเอียงของต้นไม้ ในกรณีที่ 3 ตามเอกสารประกอบคำสอน ซึ่งเป็นกรณีที่ r.left นั้นสูงกว่า r.right และ r.left.left นั้นเตี้ยกว่า r.left.right เราจะแก้ไขโดยการเรียก rotateRightChild(r.left) แล้วตามด้วย rotateLeftChild(r) จงเขียนเมท็อด Node rotateRightLeft(Node r) ซึ่งทำการแก้ไขกรณีที่ 3 ด้วยวิธีดังกล่าว <u>โดยห้ามเรียกใช้เมท็อด rotateRightChild และ rotateLeftChild</u>

```
public class AVLTree extends BSTree {
  //ไม่ได้แสดงเมท็อดหรือสมาชิกอื่น ๆ ของ AVLTree แต่สามารถเรียกใช้ได้ตามปรกติ
  public Node rotateRightLeft(Node r) {









  }
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

5. (10 คะแนน) จงเขียนเมท็อด public static String decoding(HuffmanNode hTree, BitStream bitstream) เพื่อถอดรหัสข้อมูลที่ถูกเข้ารหัสด้วยวิธี Huffman Coding เมท็อดนี้รับ hTree เป็นต้นไม้ฮัฟฟ์แมน และ bitstream เป็นรายการของบิตที่ได้มาจากการเข้ารหัสข้อมูลที่เป็นสตริง (bitstream เป็นอ็อบเจกต์ของคลาสภายในชื่อ BitStream ที่มีเมท็อดให้เรียกใช้ตามที่เขียนในโปรแกรมข้างล่าง) หมายเหตุ : ต้นไม้ฮัฟฟ์แมนในข้อนี้มีแนวคิดคล้าย (แต่ไม่เหมือน) กับคลาส HuffmanTree ที่นำเสนอในชั้นเรียน

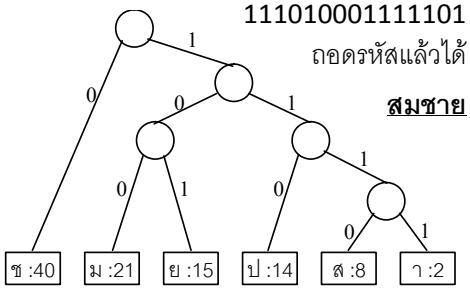
```
public class HuffmanNode implements Comparable {
  public int freq;
  public String character;
  public HuffmanNode left, right;
  public HuffmanNode(int f, String c, HuffmanNode l, HuffmanNode r) {       // constructor
    freq = f; character = c; left = l; right = r;
  }
  public boolean isLeaf() { return left == null && right == null; }
  public int compareTo(Object obj) { return freq - ((HuffmanNode) obj).freq; }
  //-------------------------------------------------------------
  public static HuffmanNode coding(int[] freq, String[] chars) {            // คืนต้นไม้ฮัฟฟ์แมน
    BinaryMinHeap h = new BinaryMinHeap();
    for (int i = 0; i < freq.length; i++)
      h.enqueue(new HuffmanNode(freq[i], chars[i], null, null));
    for (int i = 0; i < freq.length - 1; i++) {
      HuffmanNode n1 = (HuffmanNode) h.dequeue();
      HuffmanNode n2 = (HuffmanNode) h.dequeue();
      h.enqueue(new HuffmanNode(n1.freq + n2.freq, null, n2, n1));
    }
    return (HuffmanNode) h.dequeue();
  }
  //-------------------------------------------------------------
  private static class BitStream {      // เป็นอ็อบเจกต์ให้บริการเก็บรายการของบิต
    public int get(int k) { … }         // คืนค่าของ bit ที่ k มีค่า 0 หรือ 1 เท่านั้น
    public int size() { … }             // คืนจำนวนบิต
    public void remove(int k) { … }     // ลบบิตที่ k ออก
    . . .
  }
  //-------------------------------------------------------------
  public static String decoding(HuffmanNode hTree, BitStream bitstream) {
```

ตัวอย่าง

111010001111101

ถอดรหัสแล้วได้

สมชาย




```
  }
}
```

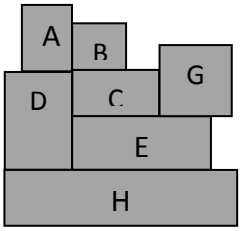
ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

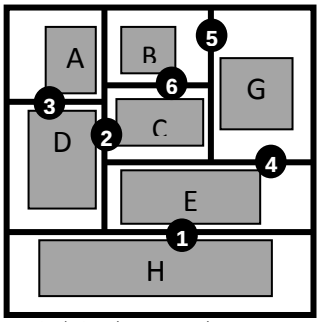
6. (10 คะแนน) เราต้องการปรับปรุงการทำงานของเมท็อด public get(Object e) ของคลาส BSTree ใหม่โดยมีเป้าหมายคือ ทำให้ปมที่ถูกค้นพบโดย get นั้นถูกย้ายตำแหน่งขึ้นไปอยู่ที่รากของต้นไม้ โดยที่ต้นไม้ยังคงเป็น binary search tree อยู่ การทำงานดังกล่าวสามารถทำได้ง่าย ๆ โดยใช้กระบวนการหมุนปมดังที่ได้ศึกษาไปในเรื่อง AVLTree โดยสามารถอธิบายขั้นตอนได้ดังนี้ เมื่อเราพบปมที่เราต้องการ กำหนดให้ X แทนปมดังกล่าว ถ้า X เป็นลูกทางซ้ายของพ่อของ X เราจะทำการ rotateLeftChild ที่ปมพ่อของ X เพื่อทำให้ X นั้นถูกย้ายขึ้นไปแทนตำแหน่งของพ่อ แต่ถ้า X เป็นลูกทางขวา เราจะทำการ rotateRightChild ที่ปมพ่อของ X ซึ่งจะทำให้ X ถูกย้ายขึ้นไปแทนตำแหน่งของพ่อเช่นเดียวกัน จงปรับปรุงเมท็อด getNode ดังต่อไปนี้ โดยให้เขียนเติมลงในพื้นที่ที่เว้นว่างไว้เท่านั้น เพื่อให้ getNode นั้นทำงานตามที่ได้กล่าวมา คำแนะนำ: ในคลาส BSTree นั้นมีเมท็อด rotateLeftChild และ rotateRightChild ให้เรียกใช้ได้อยู่แล้ว ควรจะใช้เมท็อดดังกล่าวด้วย (หมายเหตุ: ข้อความต่อไปนี้ไม่เกี่ยวข้องกับโจทย์ การปรับปรุงนี้มีเป้าหมายเพื่อทำให้ปมที่ถูกค้นนั้น เมื่อถูกค้นแล้วการค้นปมดังกล่าวจะเร็วยิ่งขึ้นในการค้นครั้งถัด ๆ ไป ซึ่งแนวคิดดังกล่าวถูกนำไปใช้ในการสร้างโครงสร้างข้อมูลที่ชื่อ Splay Tree)

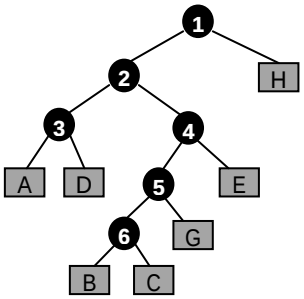
```
public Object get(Object e) {
  Node node = getNode(root, e);
  return node == null ? null : node.element;
}

Node getNode(Node r, Object e) {

  if (r == null) return null;

  int cmp = compare(e, r.element);

  if (cmp == 0) return r;

  if (cmp < 0) {



    return getNode(r.left, e);

  } else {



    return getNode(r.right, e);

  }
}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

7. (10 คะแนน) มีสี่เหลี่ยมผืนผ้าหลายรูป ปัญหาหนึ่งที่น่าสนใจคือ จะวางสี่เหลี่ยมอย่างไร ไม่ให้ซ้อนทับกัน และใช้บริเวณล้อมรอบที่เป็นรูปสี่เหลี่ยมที่มีพื้นที่น้อยสุด ๆ เราคงไม่ให้ออกแบบวิธีหาคำตอบของปัญหานี้ แต่สิ่งที่เราสนใจคือ การจัดวางสี่เหลี่ยมต่าง ๆ สามารถแทนได้ด้วย binary tree รูปซ้ายข้างล่างนี้ คือ การจัดวางรูปแบบหนึ่ง เราสามารถจินตนาการการวางแบบรูปซ้ายนี้ เสมือนการแบ่งพื้นที่การวางออกเป็นส่วน ๆ ก่อนอื่นขอเลื่อนสี่เหลี่ยมในรูปซ้ายออกจากกันเล็กน้อย เพื่อความสะดวกในการอธิบาย ดังรูปกลาง การจัดวางนี้ มาจากการเริ่มแบ่งพื้นที่ออกด้วยเส้นแบ่งแนวนอน ❶ ส่วนล่างวาง H ส่วนบนวางที่เหลือ แบ่งต่อด้วยเส้นแบ่งแนวตั้ง ❷ ได้ด้านซ้ายสำหรับ A และ D ด้านขวาสำหรับที่เหลือ แบ่งต่อสลับแนวนอน แนวตั้งเช่นนี้ ไปจนแต่ละบริเวณมีสี่เหลี่ยมเพียงรูปเดียว สามารถแทนการแบ่งนี้ได้ด้วย binary tree ดังรูปขวา







กำหนดให้คลาส BNode แทนปมต่าง ๆ ของต้นไม้ มีรายละเอียดดังแสดงทางขวานี้ ให้สังเกตว่า แต่ละปมมีความกว้างและความสูงกำกับ ถ้าเป็นใบก็เป็นสี่เหลี่ยมย่อมมีความสูงความกว้าง แต่ความกว้างความสูงของปมภายในคืออะไร ? อ๋อ มันก็คือความกว้างและความสูงของบริเวณล้อมรอบเล็กสุดที่ครอบสี่เหลี่ยมลูกหลานทั้งหมดของปมนั้น เช่น ปม ❸ ที่แทนเส้นแบ่งแนวนอน ❸ ในรูปกลาง และก็แทนบริเวณเล็กสุดที่ล้อมรอบสี่เหลี่ยม A กับ D ด้วย (อ้อ ในกรณีของปมภายใน ถ้าตัวแปรที่ชื่อ isHorizontal เป็น true แสดงว่า ปมนั้นแบ่งตามแนวนอน ถ้าเป็น false ก็แบ่งตามแนวตั้ง เช่น ปม❸ แบ่งแนวนอนจึงมี isHorizontal เป็น true)

```
class BNode {
   double width, height;
   String id;
   boolean  isHorizontal;
   BNode left, right;
}
```

ก็มาถึงสิ่งที่ต้องการให้เขียนสักที จงเติมรายละเอียดของเมท็อด getMinimumBoundingBoxArea(BNode r) ที่คืนขนาดของพื้นที่ของบริเวณที่ล้อมรอบสี่เหลี่ยมทั้งหลายของการจัดวางซึ่งแทนด้วยต้นไม้ที่มี r เป็นราก และตั้งค่า width และ height ของ r ให้ถูกต้องด้วย

```
public static double getMinimumBoundingBoxArea( BNode r ) {









}
```

ชื่อ_____นามสกุล_____หมายเลขประจำตัว_____เลขที่ใน CR58_____

8. (10 คะแนน) กำหนดให้ปัญหาการนับความถี่ของคำเป็นดังนี้ ให้มีข้อมูลหนังสือภาษอังกฤษ 1 เล่ม (ข้อมูลอยู่ในลักษณะ List ของ String โดยที่ข้อมูลแต่ละตัวใน List ดังกล่าวคือคำในหนังสือเรียงกันตั้งแต่คำแรกถึงคำสุดท้าย) เราต้องการทราบว่าในหนังสือดังกล่าวนั้น มีคำ X ปรากฏอยู่กี่ครั้ง ให้ออกแบบโครงสร้างข้อมูลชื่อ WordCount ซึ่งต้องทำงานได้ดังต่อไปนี้

   - สามารถรับข้อมูลหนังสือเข้าไปได้
   - สามารถตอบได้ว่าในข้อมูลหนังสือที่รับเข้ามานั้น มีคำ X ปรากฏอยู่กี่ครั้งโดยที่ X เป็น String ถ้าไม่มีคำนั้นอยู่เลยให้ตอบ 0

   สำหรับข้อมูลหนังสือแต่ละเล่มที่รับเข้าไปนั้น โครงสร้างข้อมูล WordCount จะถูกเรียกถามว่ามีคำว่า X อยู่หลาย ๆ ครั้งซ้ำไปซ้ำมา เพราะฉะนั้น การตอบว่ามีคำ X อยู่กี่ครั้งนั้นควรจะทำงานเร็วที่สุดเท่าที่จะทำได้

   8.1 จงออกแบบโครงสร้างข้อมูลโดยอธิบายแนวทางการทำงานพอสังเขป โดยให้ตอบไม่เกินพื้นที่ที่เว้นว่างไว้

   .....

   .....

   .....

   .....

   .....

   .....

   8.2 จงแสดงตัวอย่าง การสร้าง และวิธีการหาว่า มี X ปรากฏอยู่กี่ครั้งในหนังสือ ควรวาดรูปพร้อมคำอธิบายช่วยในการตอบคำถาม