

## CPE217 – Homework 4

### Homework: Tree Traversals using Non-Recursive Technique

Homework Due Date: 25 October 2016

Patiwet Wuttisarnwattana, Ph.D.

Department of Computer Engineering

- ☐ คำชี้แจงการส่งงาน
- ☐ ให้นักศึกษาส่งงานเข้าอีเมล [class.submissionx@gmail.com](mailto:class.submissionx@gmail.com) เท่านั้น ห้ามส่งงานเข้าเมลของอาจารย์ โปรดจำไว้ว่าอีเมลนี้อาจารย์จะไม่อ่านจนกว่าจะถึงเวลาตรวจ เพราะฉะนั้นถ้านักศึกษามีคำถามจะถามอาจารย์หรือพี่ TA ให้ส่งอีเมลเข้าอีเมลส่วนตัวอาจารย์หรือพี่ TA ตามที่ให้ไว้
- ☐ ในการส่งงานให้ระบุในช่อง Subject ของ email ให้เขียน pattern ดังต่อไปนี้ [CPE217][HWx][core person ID] ตรง x ให้ระบุว่าเป็นการบ้านที่เท่าไรและตรง core person ID ให้ระบุรหัสนักศึกษาที่เป็น core person ตัวอย่างเช่น [CPE217][HW4][580610615] หาก core person คือ กนกวรรณ
- ☐ แต่ละกลุ่ม ควรให้ core person เป็นคนส่งงาน และในเนื้อความอีเมล ให้ระบุ รหัสประจำตัวนักศึกษาของทุกคนที่เป็นสมาชิกในกลุ่ม หาก core person ไม่สามารถส่งงานได้ ให้สมาชิกคนใดก็ได้ส่งงานแทน แต่ core person id ต้องเป็นรหัสเดิมทุกครั้ง
- ☐ โค้ดของคุณต้องมีคอมเมนต์ (comment) เพื่ออธิบายว่าโค้ดดังที่เห็นอยู่นี้ทำงานอะไร หรือ if นี่ทำตรวจสอบอะไร หากกลุ่มไหนไม่มีคอมเมนต์ในโค้ดจะไม่ได้รับการตรวจ การเขียนคอมเมนต์ไม่ต้องเขียนแบบละเอียดยิบก็ได้ เท่าที่คุณต้องการให้ผู้ตรวจทราบก็พอ
- ☐ อาจารย์เปลี่ยนใจละ ไม่ต้องส่งคะแนน contribution ของเพื่อนในกลุ่ม หากกลุ่มไหนทนไม่ไหวและอยากรายงานความประพฤติของเพื่อนว่าเขาเปรี้ยบ ไม่ช่วยทำงาน ให้ส่งอีเมลแจ้งอาจารย์เป็นกรณีไป
- ☐ งานที่ส่งต้องประกอบด้วย Zip file ของ src folder ที่สามารถกด F6 รันได้เลย หากมี compile error หรือ runtime exception งานของนักศึกษาจะไม่ได้รับการตรวจ
- ☐ สามารถส่งการบ้านซ้ำได้ แต่หักคะแนนวันละ 10%

การบ้านนี้ให้นักศึกษา implement Tree Traversals using Non-recursion Technique โดยใช้ Java โดยให้มีคลาสดังต่อไปนี้

1. ให้สร้าง class ชื่อว่า Queue โดย class นี้ มีคุณสมบัติของ Queue ADT ตามที่ได้เรียนในห้องเรียน
2. ให้สร้าง class ชื่อว่า Stack โดย class นี้ มีคุณสมบัติของ Stack ADT ตามที่ได้เรียนในห้องเรียน
3. ให้ Stack และ Queue สามารถบรรจุ objects ของ class Node โดย class Node นี้มีคุณสมบัติของการเป็น Binary Tree
  - a. ให้ Node แต่ละ Node สามารถบรรจุ data (key) ได้ค่า ๆ หนึ่ง โดยให้เป็นตัวแปรชนิด integer
  - b. ให้ Node แต่ละ Node สามารถต่อกันเพื่อเป็นโครงสร้างข้อมูล Binary Tree ตามที่ได้เรียนในห้องเรียน
  - c. สมาชิกของ class Node ควรที่จะมี reference ชี้ไปยัง ลูกคนซ้าย (left child) และลูกคนขวา (right child)
  - d. การบ้านนี้กำหนดให้ ไม่มี parent reference/pointer (ทราบลูก แต่ไม่ทราบแม่)
4. ให้ class Node มี 1 Constructor คือ Node(int data) ซึ่งทำหน้าที่ กำหนดค่าเริ่มต้นของ key จาก data
5. ในการบ้านนี้ กำหนดให้คุณใช้ Circular Array ในการ implement Queue ให้ Queue มีฟังก์ชันดังต่อไปนี้
  - a. public void enqueue(Node node) ทำหน้าที่ enqueue Node ตามที่ได้เรียนในห้อง  
☐ กำหนดให้ Circular Array สามารถบรรจุข้อมูลได้เต็ม capacity เช่น capacity = 10 ข้อมูลที่สามารถบรรจุได้คือ 10 พอดี หากข้อมูลถึง 11 เข้ามา ให้แจ้งว่า “Queue Overflow!!!”
  - b. public Node dequeue() ทำหน้าที่ dequeue Node ตามที่ได้เรียนในห้อง  
☐ หากทำการ dequeue ในขณะที่ Q ว่างอยู่ให้แจ้งว่า “Queue Underflow!!!”
  - c. public void printQueue() ทำหน้าที่ แสดงว่าปัจจุบันนี้มีข้อมูลอะไรบรรจุอยู่ใน Queue บ้าง pattern การแสดงออกทาง Console ให้เป็นไปตามที่เห็นใน (ดังตัวอย่างด้านล่าง) เริ่มต้นด้วย [Front] ลงท้ายด้วย [Back]
  - d. public void printCircularIndices() ทำหน้าที่ แสดงว่าปัจจุบันนี้ front index กับ back index (ตามหลักการที่เรียนในห้อง)
6. ในการบ้านนี้ กำหนดให้คุณใช้ Array of Nodes ในการ implement Stack ให้ Stack มีฟังก์ชันดังต่อไปนี้
  - a. public void push(Node node) ทำหน้าที่ push Node ตามที่ได้เรียนในห้อง  
☐ หากข้อมูลที่ใส่เข้ามาใหม่ เกิน capacity ให้แจ้งว่า “Stack Overflow!!!”
  - b. public Node pop() ทำหน้าที่ pop Node ตามที่ได้เรียนในห้อง  
☐ หากทำการ pop ในขณะที่ Stack ว่างอยู่ให้แจ้งว่า “Stack Underflow!!!”
  - c. public void printStack() ทำหน้าที่ แสดงว่าปัจจุบันนี้มีข้อมูลอะไรบรรจุอยู่ใน Stack บ้าง pattern การแสดงออกทาง Console ให้เป็นไปตามที่เห็นใน (ดังตัวอย่างด้านล่าง) เริ่มต้นด้วย [Bottom] ลงท้ายด้วย [Top]
7. การบ้านนี้อาจารย์ได้เพิ่มคุณสมบัติพิเศษของ Node คือ คุณสามารถที่จะพิมพ์แผนภาพต้นไม้ออกมาทาง Console ได้ เพียงแค่เรียกใช้ฟังก์ชัน public void printTree() (ดังตัวอย่างด้านล่าง) คุณไม่ต้องเขียนฟังก์ชันนี้เอง แต่คุณต้องนำ class พิเศษของอาจารย์เข้าไปด้วย โดยให้คุณทำตาม Step ดังต่อไปนี้
  - a. ให้คุณนำไฟล์ BTreePrinter.java เข้าไปอยู่ในโปรเจกต์และ package ปัจจุบันของคุณ
  - b. ให้คลาส Node ของคุณ สืบทอดคุณสมบัติ (OOP inheritance) ของคลาสที่มีชื่อว่า BTreePrinter (คุณควรที่จะรู้ว่าต้องใช้คำสั่งอะไรใน Java เพื่อ class หนึ่ง ๆ จะทำการสืบทอดคุณสมบัติของ class อีกอันหนึ่ง)

- c. ให้คลาส Node ของคุณ มีฟังก์ชันที่ชื่อว่า `public void printTree()` โดยหน้าที่ของฟังก์ชันนี้คือการเรียกใช้ฟังก์ชัน `protected void printTree(Node node)` (ที่คุณสืบทอดมาจาก `BTreePrinter`) อีกทีหนึ่ง พารามิเตอร์ `node` ที่ส่งเข้าไป คือ `root node` ของแผนภาพต้นไม้
  - d. ให้คุณคิดว่า class `BTreePrinter` เป็นเครื่องมือในการแสดงผล คุณไม่จำเป็นต้องรู้ว่า class `BTreePrinter` ทำงานอย่างไร รู้แต่ว่าติดตั้งอย่างไรและใช้งานอย่างไรก็พอ
  - e. ให้คุณทำการสร้าง trees (class `Node`) 2 ต้น ผ่าน function `constructTree1()` และ `constructTree2()` แล้วให้แสดงแผนภาพต้นไม้โดยการเรียกใช้ฟังก์ชัน `printTree()` ตามตัวอย่างการทำงาน 10.1 และ 10.2
8. ให้คุณทำการ implement Breadth-first Traversal โดยใช้ Queue ตามที่คุณเรียนในห้อง โดยทำเป็นฟังก์ชันที่เป็นหนึ่งของ class `Node` โดยมี prototype ดังต่อไปนี้
- a. `public void printBFT()`
  - b. ให้ pattern การพิมพ์ออกทาง console ให้เป็นไปดังตัวอย่างด้านล่าง เริ่มต้นด้วยคำว่า “BFT node sequence [ ” ลงท้ายด้วย “]”
9. ให้คุณทำการ implement PreOrder Depth-first Traversal โดยใช้ Stack ตามที่คุณเรียนในห้อง โดยทำเป็นฟังก์ชันที่เป็นส่วนหนึ่งของ class `Node` โดยมี prototype ดังต่อไปนี้
- a. `public void printDFT()`
  - b. ให้ pattern การพิมพ์ออกทาง console ให้เป็นไปดังตัวอย่างด้านล่าง เริ่มต้นด้วยคำว่า “DFT node sequence [ ” ลงท้ายด้วย “]”
10. ตัวอย่างการทำงาน

Java code
<pre> public static void main(String[] args) {     Node tree = constructTree1();     tree.printTree(); } </pre>
Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)
<pre>       3      /\     /  \    /    \   /      \  /        \ 7          5  /\        \ /  \      \ 2  6    9   /\    /  1 8 4 </pre>

Java code
<pre> public static void main(String[] args) {     Node tree = constructTree2();     tree.printTree(); } </pre>
Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)
<pre>       1      /\     /\     /\    /\   /\  /\  /\  /\  2   3  /\  \ /\   \ /\   \ /\   \ 4  5   6    /\  /    /\ /   7 8  9      \     10 </pre>

Java code
<pre> public static void main(String[] args) {     Stack s = new Stack(4);     s.pop();     s.push(new Node(5));     s.push(new Node(6));     s.push(new Node(7));     s.push(new Node(8)); } </pre>

```
s.printStack();  
s.push(new Node(9));  
System.out.println(s.pop().data);  
System.out.println(s.pop().data);  
System.out.println(s.pop().data);  
s.printStack();  
}
```

### Output

Stack Underflow!!!  
[Bottom] 5 6 7 8 [Top]  
Stack Overflow!!!  
8  
7  
6  
[Bottom] 5 [Top]

### Java code

```
public static void main(String[] args) {  
    Queue q = new Queue(4);  
    q.dequeue();  
    q.enqueue(new Node(5));  
    q.enqueue(new Node(6));  
    q.enqueue(new Node(7));  
    q.enqueue(new Node(8));  
    q.printQueue();  
    q.enqueue(new Node(9));  
    System.out.println(q.dequeue().data);  
    System.out.println(q.dequeue().data);  
    System.out.println(q.dequeue().data);  
    q.printQueue();  
}
```

### Output

Queue Underflow!!!  
[Front] 5 6 7 8 [Back]

Queue Overflow!!!

5

6

7

[Front] 8 [Back]

#### Java code

```
public static void main(String[] args) {  
    Queue q = new Queue(4);  
    q.printCircularIndices();  
    q.enqueue(new Node(5));  
    q.enqueue(new Node(6));  
    q.printCircularIndices();  
    q.enqueue(new Node(7));  
    q.enqueue(new Node(8));  
    q.printCircularIndices();  
    q.printQueue();  
    System.out.println(q.dequeue().data);  
    q.printCircularIndices();  
    System.out.println(q.dequeue().data);  
    q.printCircularIndices();  
    System.out.println(q.dequeue().data);  
    q.printCircularIndices();  
    q.enqueue(new Node(9));  
    q.enqueue(new Node(10));  
    q.enqueue(new Node(11));  
    q.printQueue();  
}
```

#### Output

Front index = 0 Back index = 0

Front index = 0 Back index = 2

Front index = 0 Back index = 0

[Front] 5 6 7 8 [Back]

5

Front index = 1 Back index = 0

6

Front index = 2 Back index = 0

7

Front index = 3 Back index = 0

[Front] 8 9 10 11 [Back]

#### Java code

```
public static void main(String[] args) {  
    Node tree = constructTree1();  
    tree.printTree();  
    tree.printBFT();  
    tree.printDFT();  
}
```

#### Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)

```
    3  
   /\n  /\n /\n/\n/\n7  5  
\  \n/\  \n2 6  9  
  /\  /\n 18 4
```

BFT node sequence [ 3 7 5 2 6 9 1 8 4 ]

DFT node sequence [ 3 7 2 6 1 8 5 9 4 ]

11. โปรดใช้ Starter code ที่อาจารย์แนบให้