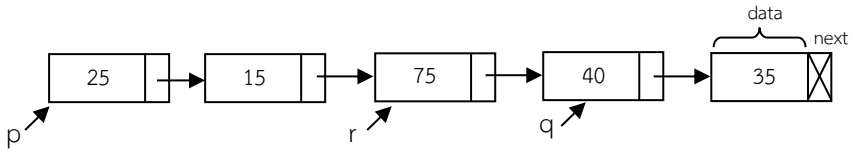
แบบฝึกหัดเสริม

วิชา Data Structures and Algorithms

(บทที่ 2 – 6)

บทที่ 2 : Array and Linked List

1. กำหนดให้มี Singly linked list ดังนี้



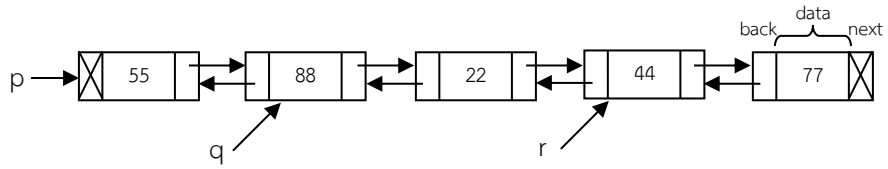
จงวาดภาพผลลัพธ์เมื่อประมวลผลคำสั่ง 1) – 7) ทั้งหมดแล้ว โดยผลจากการทำคำสั่ง 1) จะเป็นอินพุทของการทำคำสั่ง 2) และผลการทำ 2) จะเป็นอินพุทของการทำ 3) ไปเรื่อยๆ

1) p.next.next = q.next
2) r.next.next = p
3) q = q.next.next
4) q.next.data = 50
5) p = r
6) r = p.next.next
7) r.data = 10

จากผลลัพธ์ข้างต้น จงหาผลลัพธ์ของ

```
loop (p.next != null)
        print p.data
        p = p.next
end loop
```

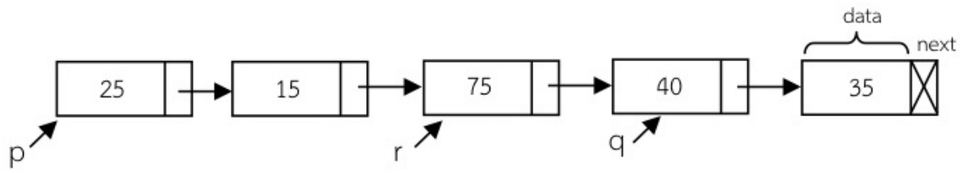
2. กำหนดให้มี Doubly linked list ดังนี้



จงวาดภาพผลลัพธ์เมื่อประมวลผลคำสั่ง 1) – 7) ทั้งหมดแล้ว โดยผลจากการทำคำสั่ง 1) จะเป็นอินพุทของการทำคำสั่ง 2) และผลการทำ 2) จะเป็นอินพุทของการทำ 3) ไปเรื่อยๆ

1) q.next = r
2) r.back.back.next.data = 99
3) r.next.next = p
4) q.back.back = r.next
5) r.back.back.next = NULL
6) r.back = q.next
7) p.data = r.next.data

1. กำหนดให้มี Singly linked list ดังนี้

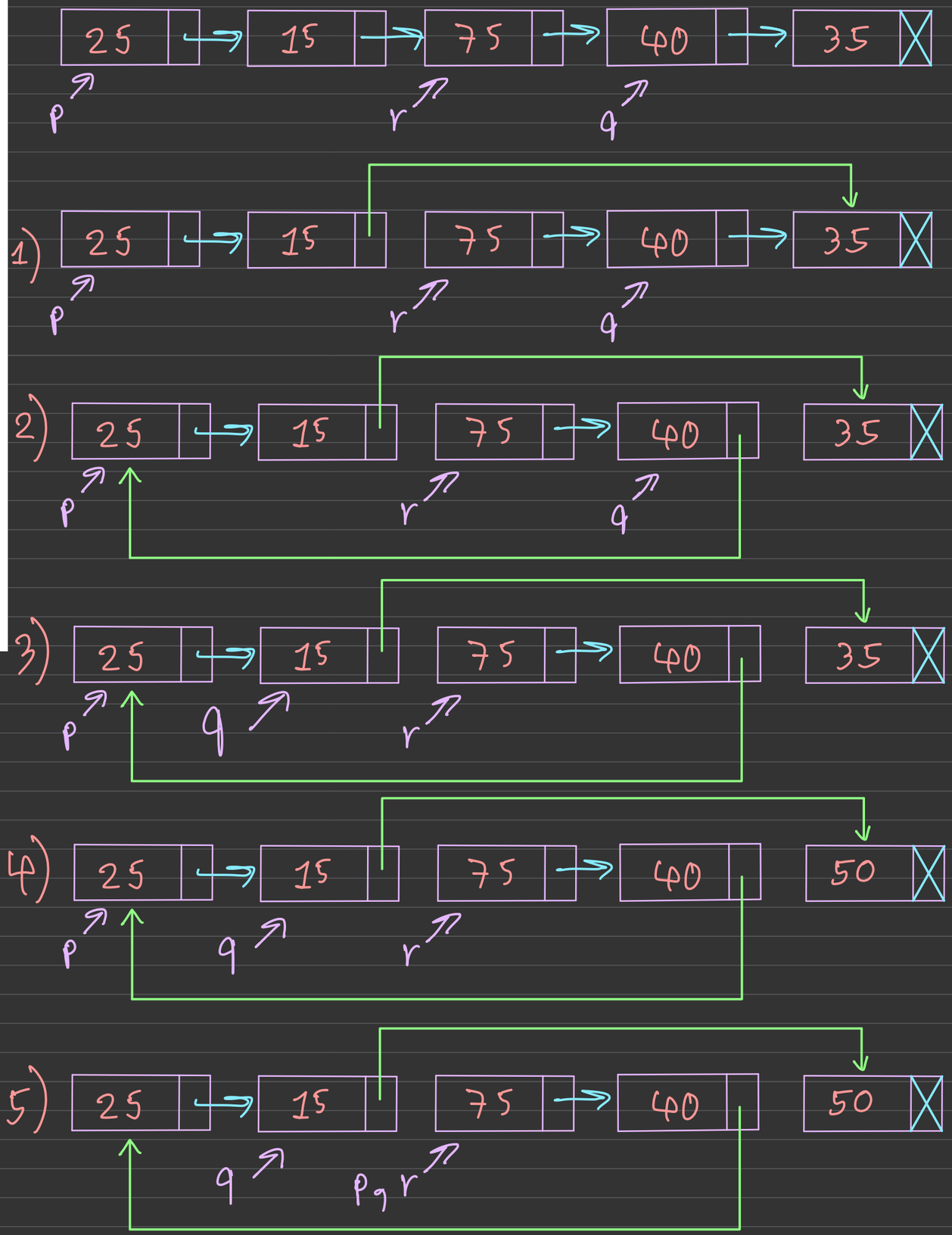


จงวาดภาพผลลัพธ์เมื่อประมวลผลคำสั่ง 1) – 7) ทั้งหมดแล้ว โดยผลจากการทำคำสั่ง 1) จะเป็นอินพุทของการทำคำสั่ง 2) และผลการทำ 2) จะเป็นอินพุทของการทำ 3) ไปเรื่อยๆ

1) p.next.next = q.next
2) r.next.next = p
3) q = q.next.next
4) q.next.data = 50
5) p = r
6) r = p.next.next
7) r.data = 10

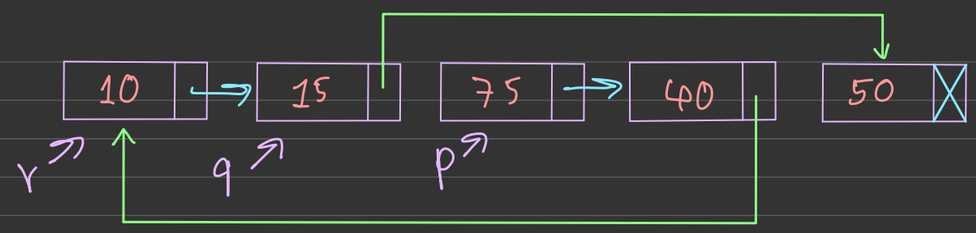
จากผลลัพธ์ข้างต้น จงหาผลลัพธ์ของ

```
loop (p.next != null)
        print p.data
        p = p.next
end loop
```



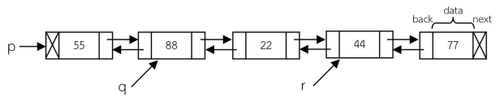
จากผลลัพธ์ข้างต้น จงหาผลลัพธ์ของ

```
loop (p.next != null)
        print p.data
        p = p.next
end loop
```



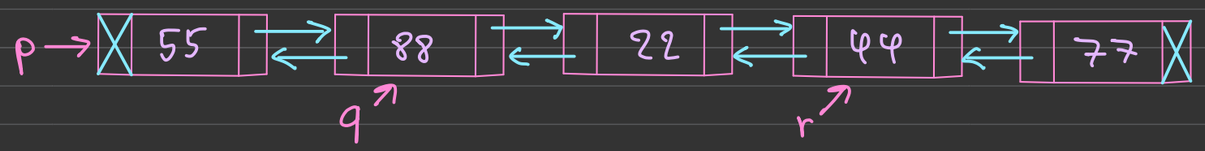
75 40 10 15 50

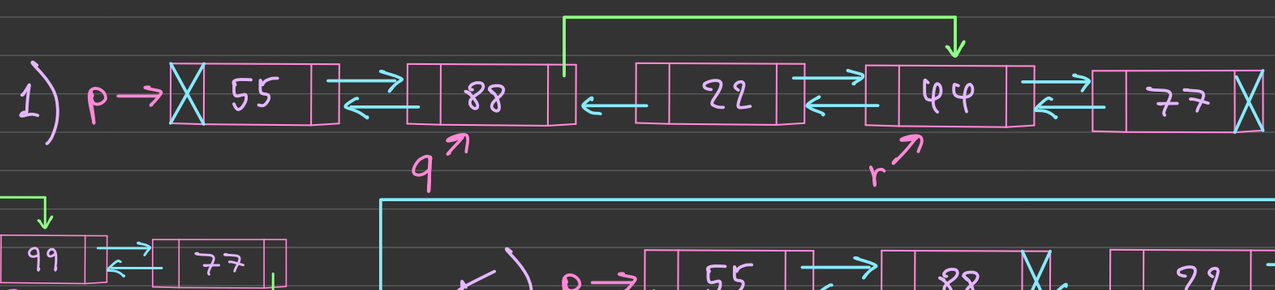
2. กำหนดให้มี Doubly linked list ดังนี้

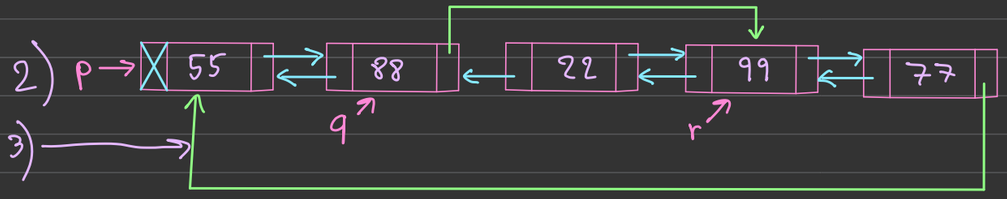


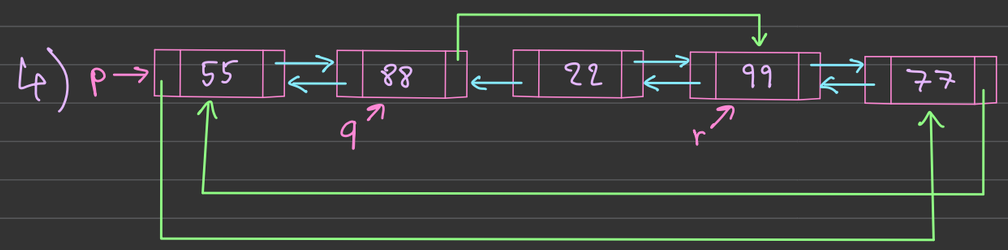
จงวาดภาพผลลัพธ์เมื่อประมวลผลคำสั่ง 1) – 7) ทั้งหมดแล้ว โดยผลจากการทำคำสั่ง 1) จะเป็นอินพุทของการทำคำสั่ง 2) และผลการทำ 2) จะเป็นอินพุทของการทำ 3) ไปเรื่อยๆ

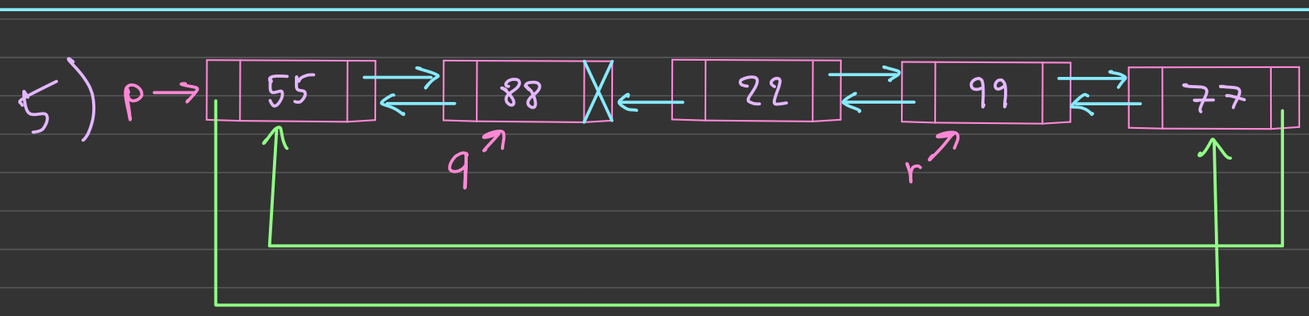
1) q.next = r
2) r.back.back.next.data = 99
3) r.next.next = p
4) q.back.back = r.next
5) r.back.back.next = NULL
6) r.back = q.next
7) p.data = r.next.data

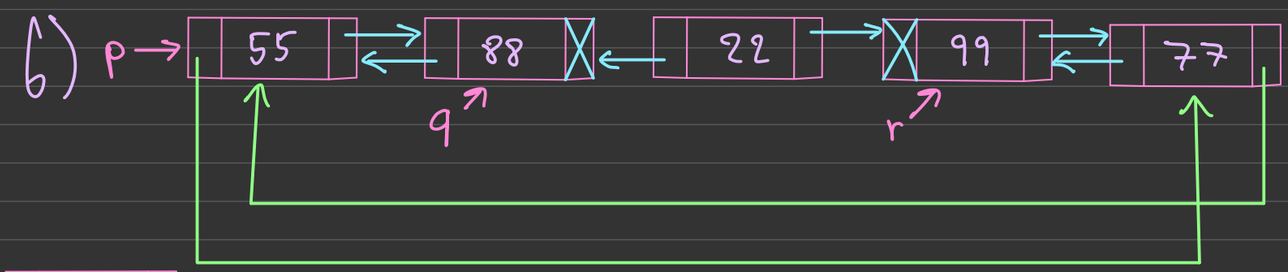


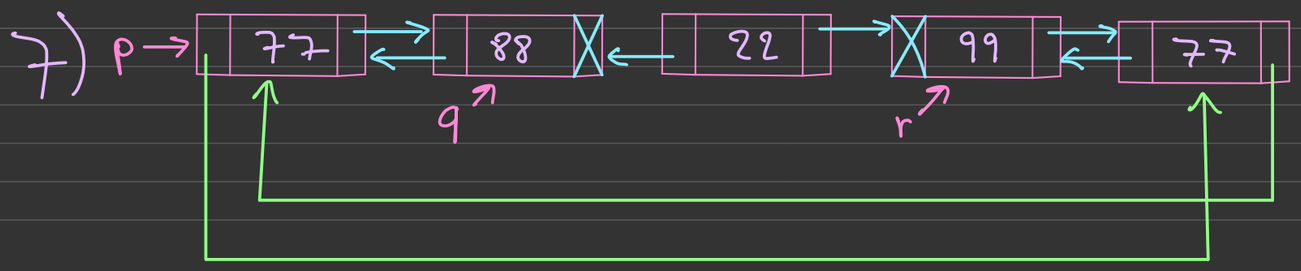












① 99 77 77

② 88 77 77

จากผลลัพธ์ข้างต้น จงหาผลลัพธ์ของ

```
loop (r.next != null)
        print r.data
        r = r.next
end loop
```

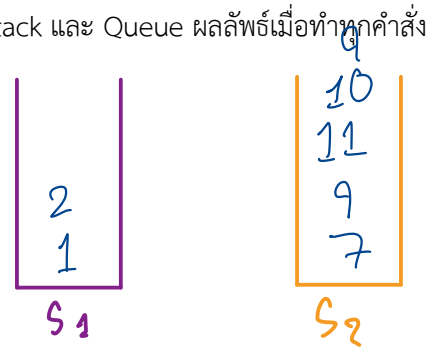
จากผลลัพธ์ข้างต้น จงหาผลลัพธ์ของ

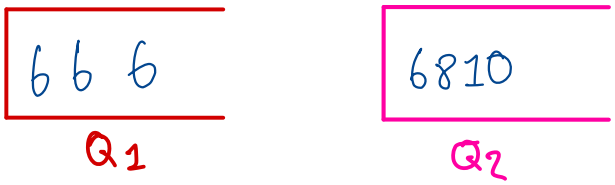
```
loop (q.back != null)
        print q.data
        q = q.back
end loop
```

บทที่ 3 : Stack and Queue

1. จงประมวลผลคำสั่งต่อไปนี้ พร้อมวาดภาพ Stack และ Queue ผลลัพธ์เมื่อทำทุกคำสั่งเสร็จแล้ว

```
S1 = createStack()
S2 = createStack()
Q1 = createQueue()
Q2 = createQueue()
for (i=1 to 5)
        pushStack(S1, i)
end for
for (i=6 to 10)
        enqueue(Q1, i)
end for
loop (Q1 is not empty)
        dequeue(Q1, x)
        if (x mod 2 == 0)
                enqueue(Q2, x)
        else
                pushStack(S2, x)
        end if
end loop
for (i=1 to 3)
        popStack(S1, x)
        queueFront(Q2, y)
        pushSatck(S2, x+y)
        enqueue(Q1, y)
end for
```





บทที่ 4 : Binary Tree, Binary Search Tree

1. จงวาดรูป Binary Tree เมื่อมีลำดับการท่องเข้าไปในต้นไม้ดังนี้

   Preorder : H C F G B J E A L K D I

   Inorder : F B G J C H A K L D E I
2. จาก Binary tree ในข้อ 1 มีลำดับการท่องต้นไม้แบบ Postorder และ Breadth-first เป็นอย่างไร
3. จงวาดรูป Binary Search Tree (BST) (เริ่มต้นเป็น Empty tree) โดยแทรกข้อมูลเข้าตามลำดับ ดังนี้

   80 90 120 70 15 85 60 40 100 55 65 25 110
4. จงลบโหนด 80 ออกจาก BST ผลลัพธ์ของข้อ 3 แล้วแสดง BST ผลลัพธ์หลังจากการลบโหนด

บทที่ 5 : Expression Tree and AVL Tree

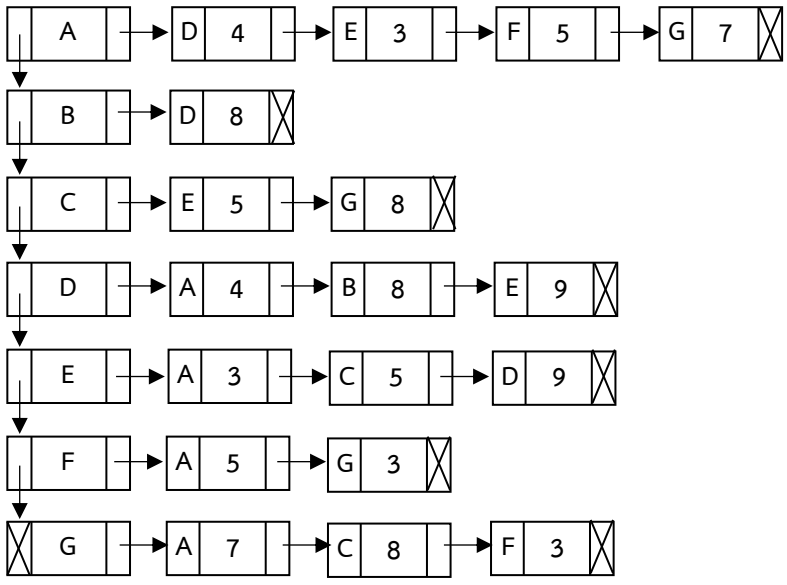
1. จงเขียน Expression Tree สำหรับ Prefix expression ดังนี้

   + / C G * - A E + D / B F
2. จงเขียน Infix และ Postfix expression จาก Expression Tree ข้อ 1
3. จงวาดรูป AVL Tree (เริ่มต้นเป็น Empty tree) โดยแทรกข้อมูลเข้าตามลำดับ ดังนี้

   80 90 120 70 15 85 60 40 100 55 65 25 110
4. จงลบโหนด 80 ออกจาก AVL ผลลัพธ์ของข้อ 3 แล้วแสดง AVL ผลลัพธ์หลังจากการลบโหนด

บทที่ 6 : Graph

กำหนดให้ กราฟ G1 มี Adjacency List ดังนี้

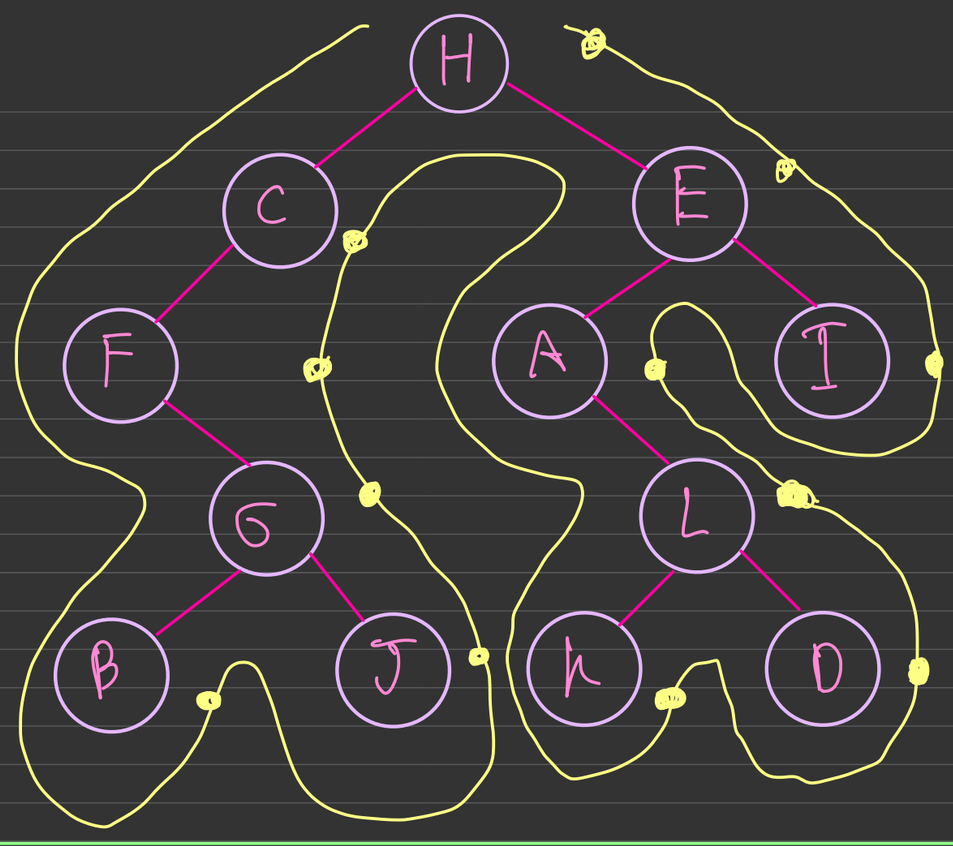


1. จงวาดรูปกราฟ G1
2. จงเขียน Minimum Spanning Tree ของกราฟนี้
3. จงเขียน Shortest Path จาก Vertex G ไปยัง Vertex อื่นๆ

1. จงวาดรูป Binary Tree เมื่อมีลำดับการท่องเข้าไปในต้นไม้ดังนี้

Preorder : H C F G B J E A L K D I

Inorder : F B G J C H A K L D E I



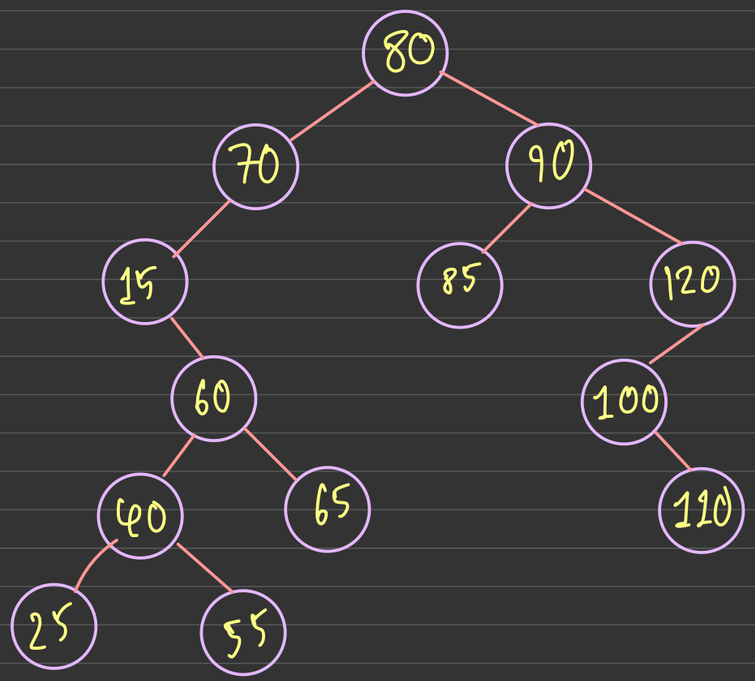
2. จาก Binary tree ในข้อ 1 มีลำดับการท่องต้นไม้แบบ Postorder และ Breadth-first เป็นอย่างไร

Post: B J G F C K D L A I E H

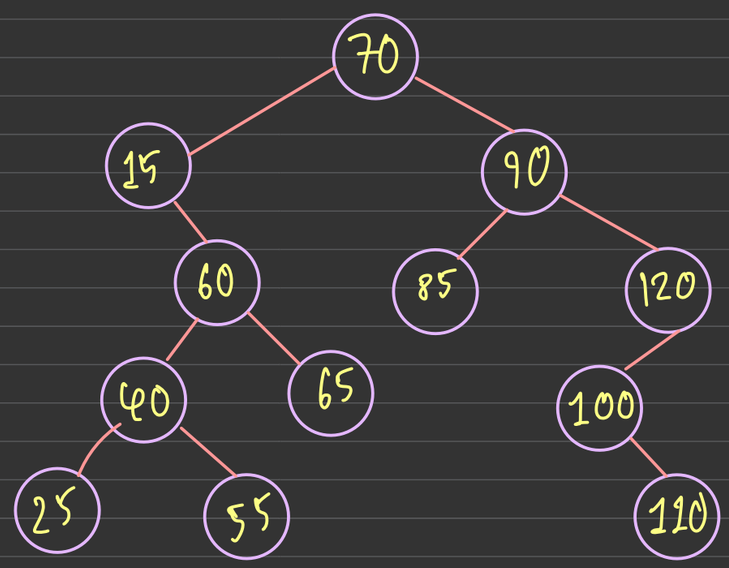
Breadth : H C E F A I G L B J K D

3. จงวาดรูป Binary Search Tree (BST) (เริ่มต้นเป็น Empty tree) โดยแทรกข้อมูลเข้าตามลำดับ ดังนี้

80 90 120 70 15 85 60 40 100 55 65 25 110



4. จงลบโหนด 80 ออกจาก BST ผลลัพธ์ของข้อ 3 แล้วแสดง BST ผลลัพธ์หลังจากการลบโหนด



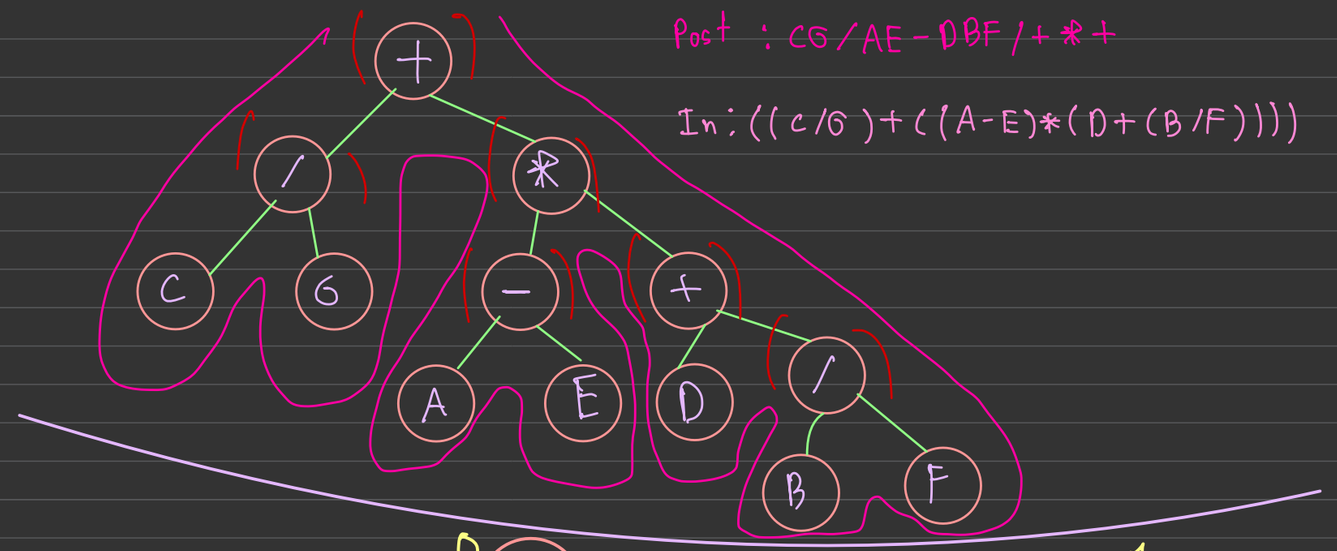
บทที่ 5 : Expression Tree and AVL Tree

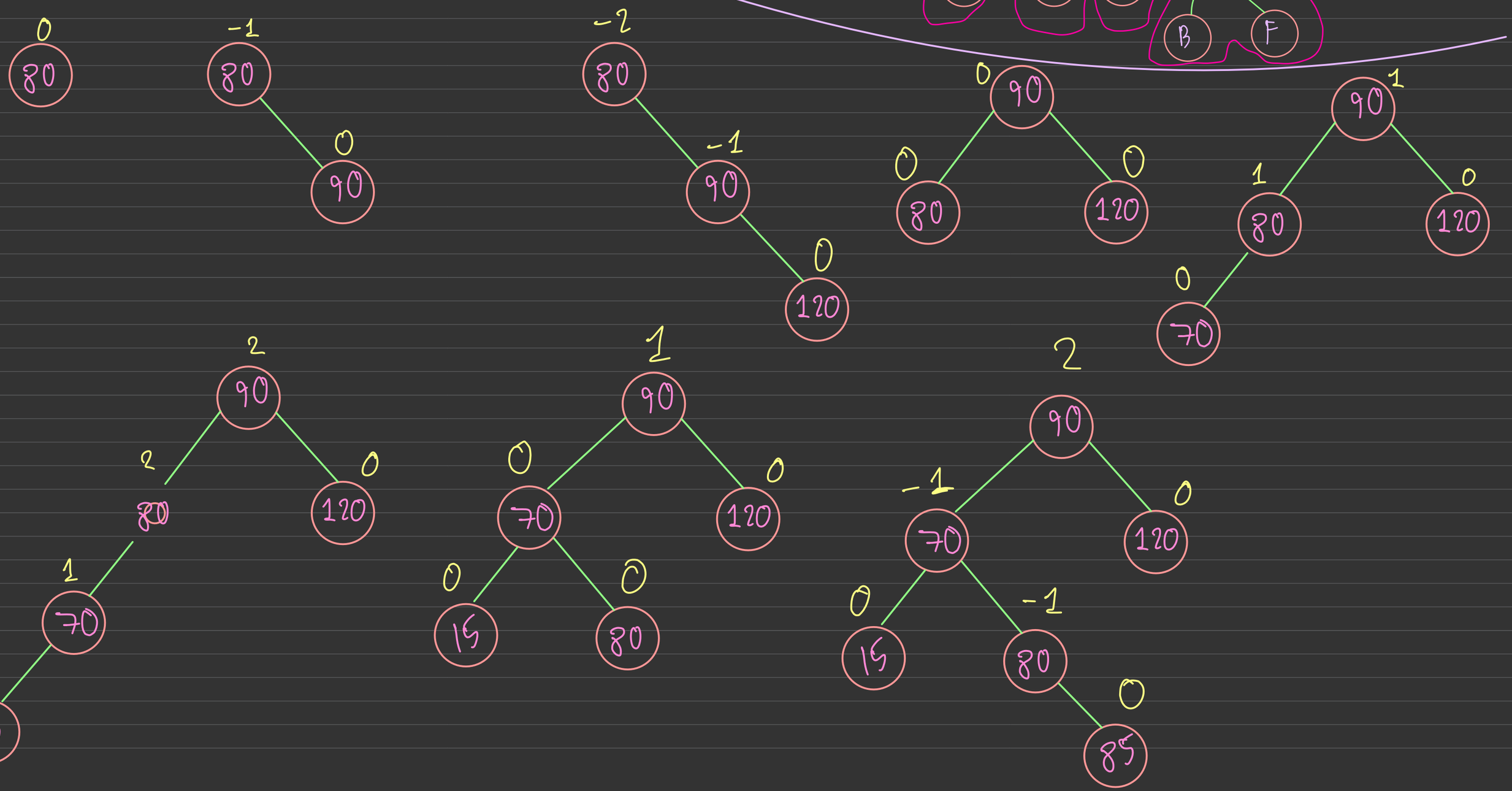
1. จงเขียน Expression Tree สำหรับ Prefix expression ดังนี้

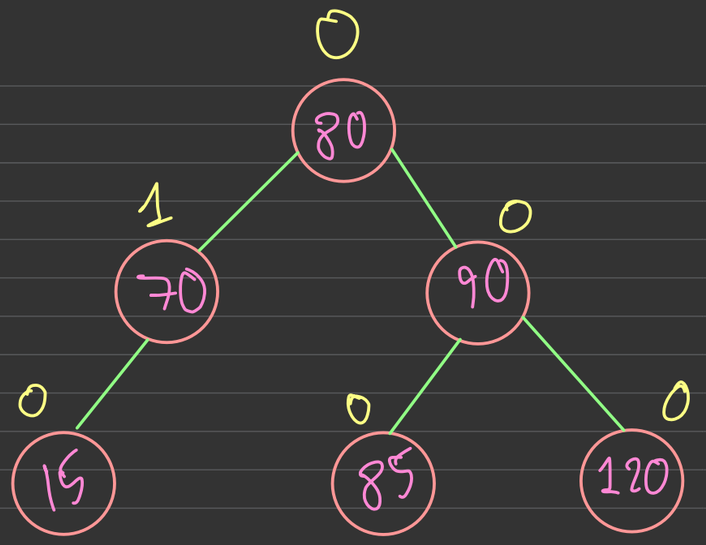
   + / C G * - A E + D / B F

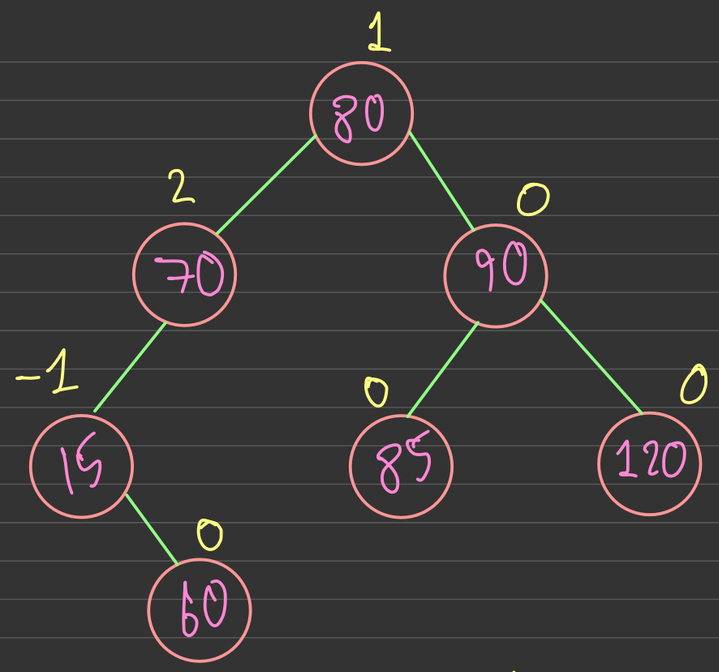
2. จงเขียน Infix และ Postfix expression จาก Expression Tree ข้อ 1
3. จงวาดรูป AVL Tree (เริ่มต้นเป็น Empty tree) โดยแทรกข้อมูลเข้าตามลำดับ ดังนี้

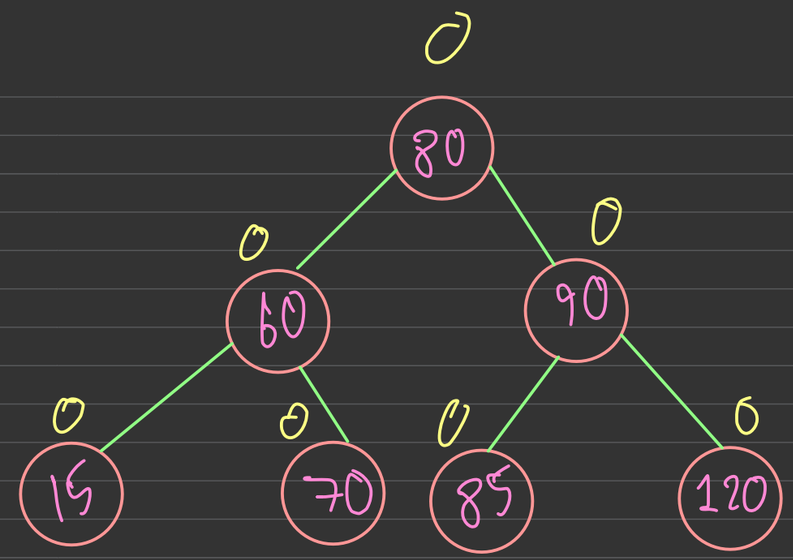
   80 90 120 70 15 85 60 40 100 55 65 25 110
4. จงลบโหนด 80 ออกจาก AVL ผลลัพธ์ของข้อ 3 แล้วแสดง AVL ผลลัพธ์หลังจากการลบโหนด

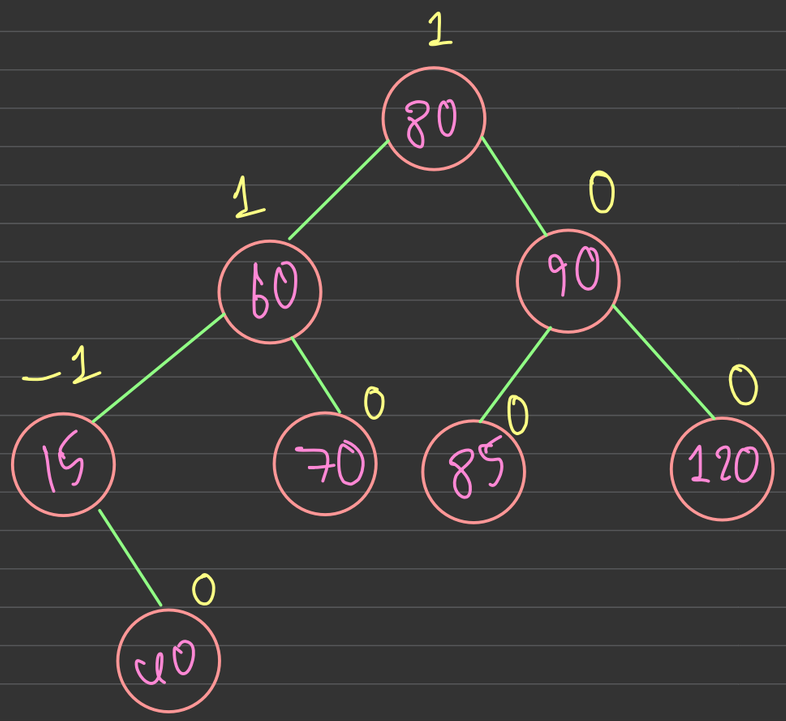


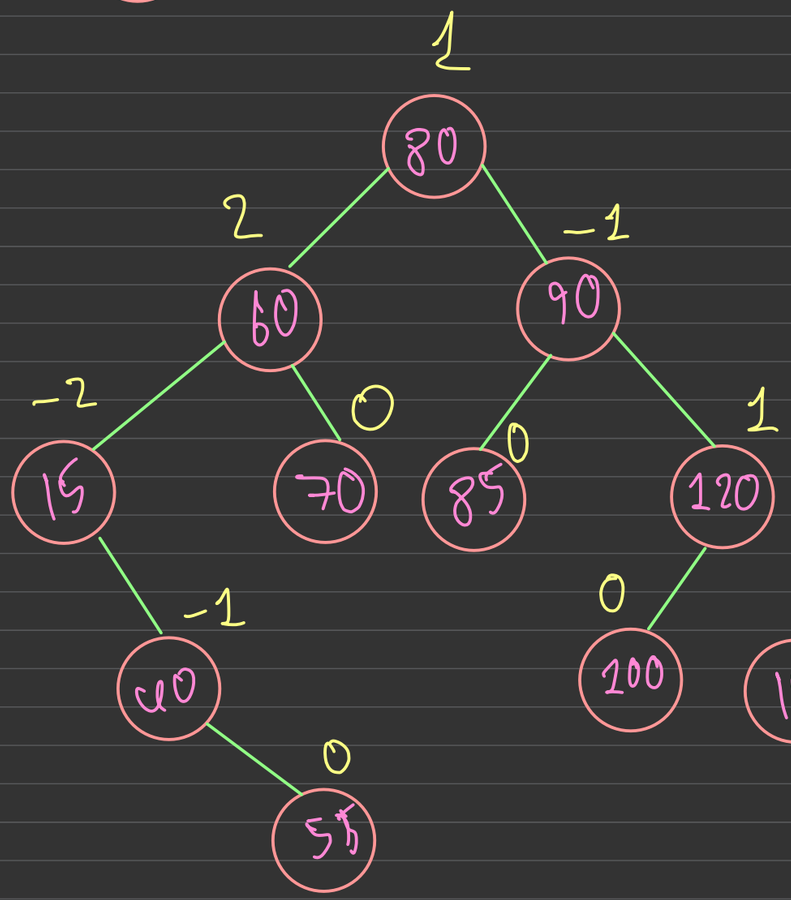


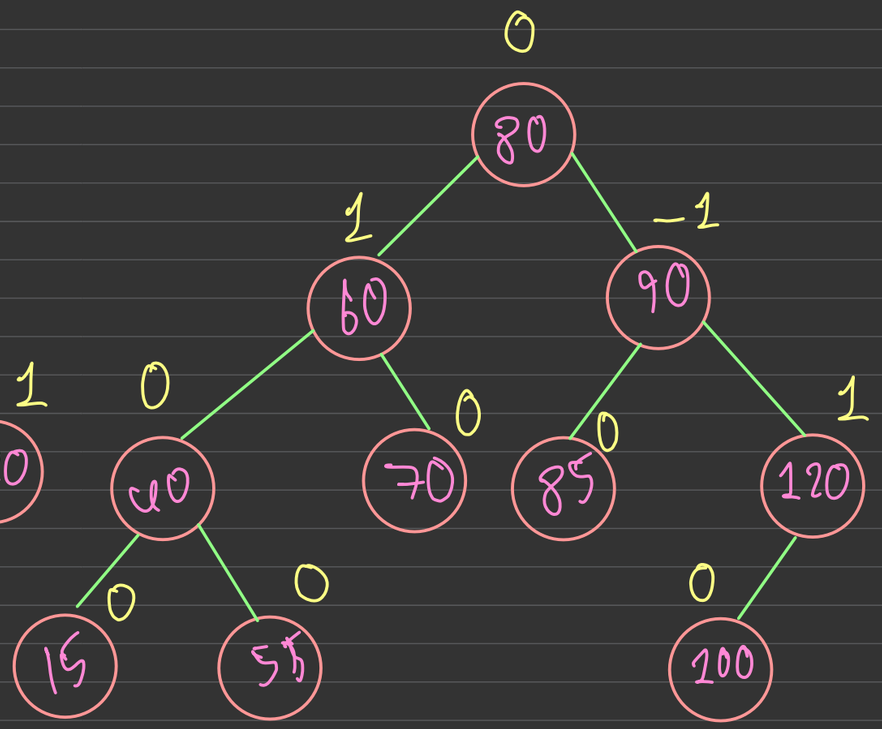

0
80
1
70
0
90
0
15
0
85
0
120

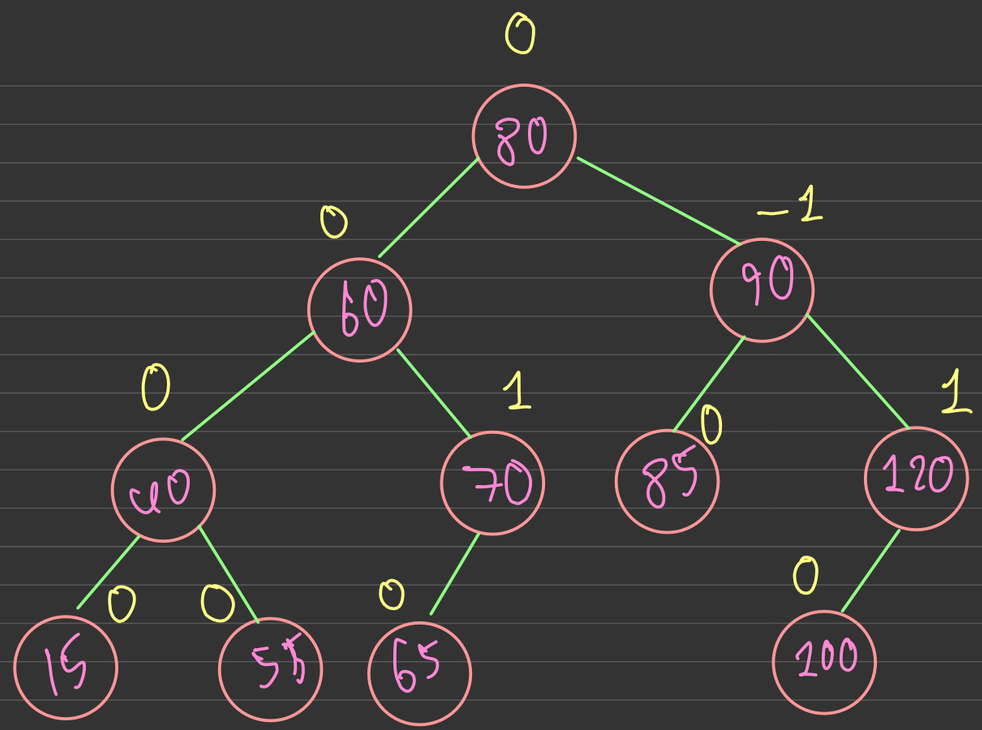

1
80
2
70
0
90
-1
15
0
85
0
120
0
60

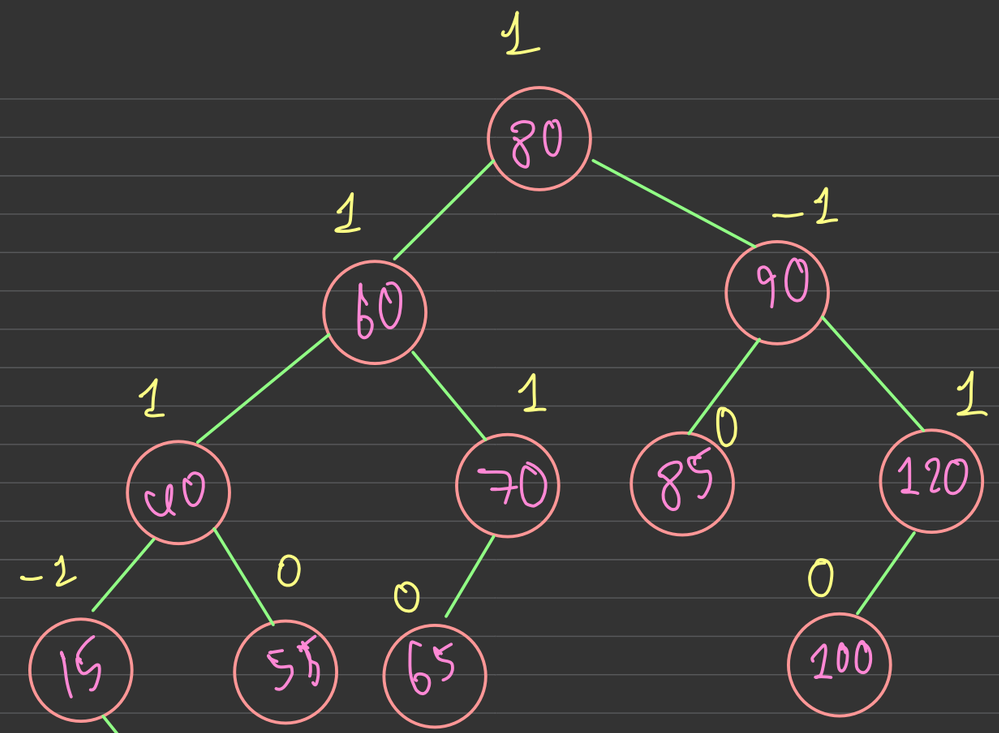

0
80
0
60
0
90
0
15
0
70
0
85
0
120

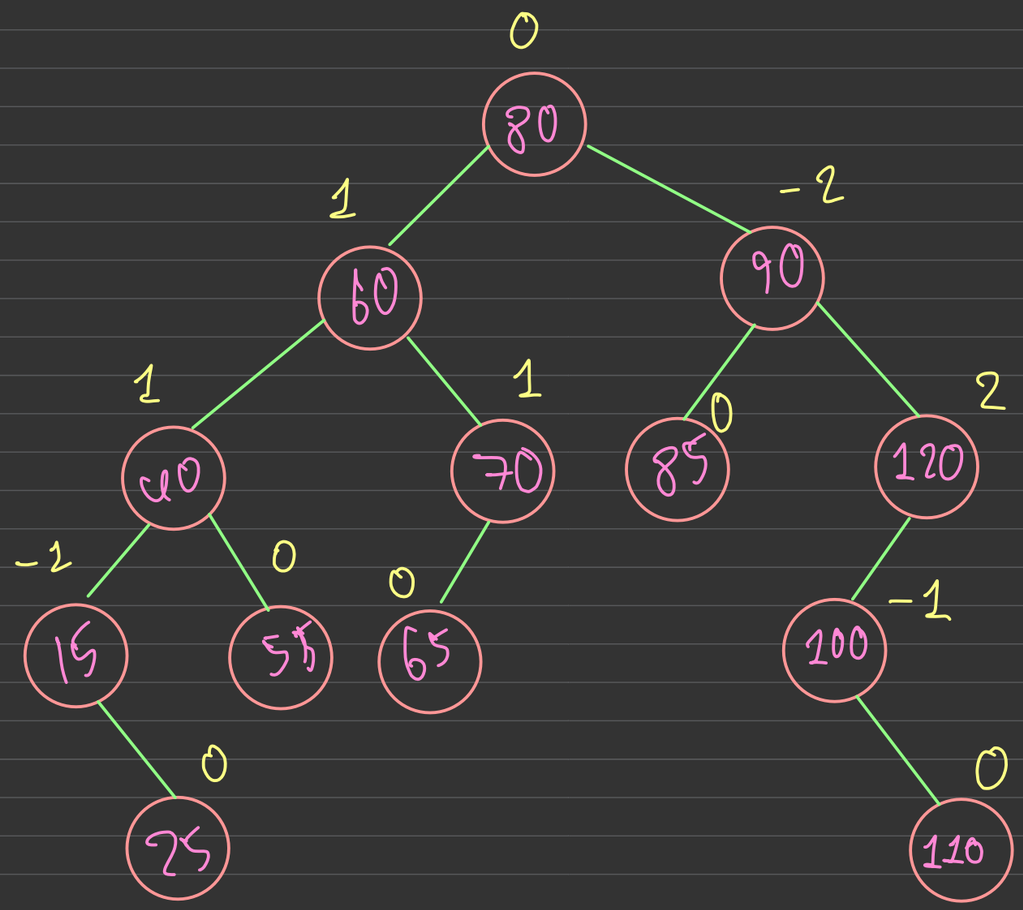

1
80
1
60
0
90
-1
15
0
70
0
85
0
120
0
40

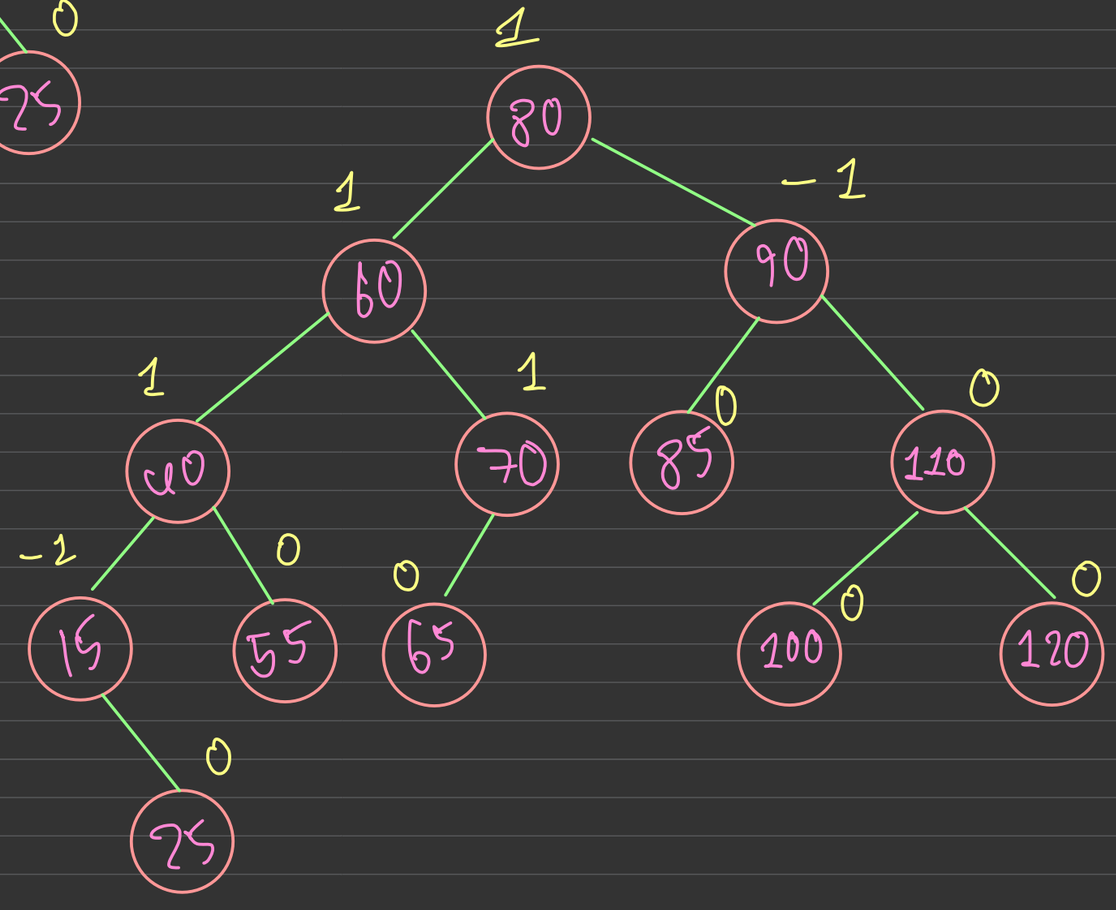

1
80
2
60
-1
90
-2
15
0
70
0
85
1
120
-1
40
0
55
0
100


0
80
1
60
-1
90
0
40
0
70
0
85
1
120
0
15
0
55
0
100

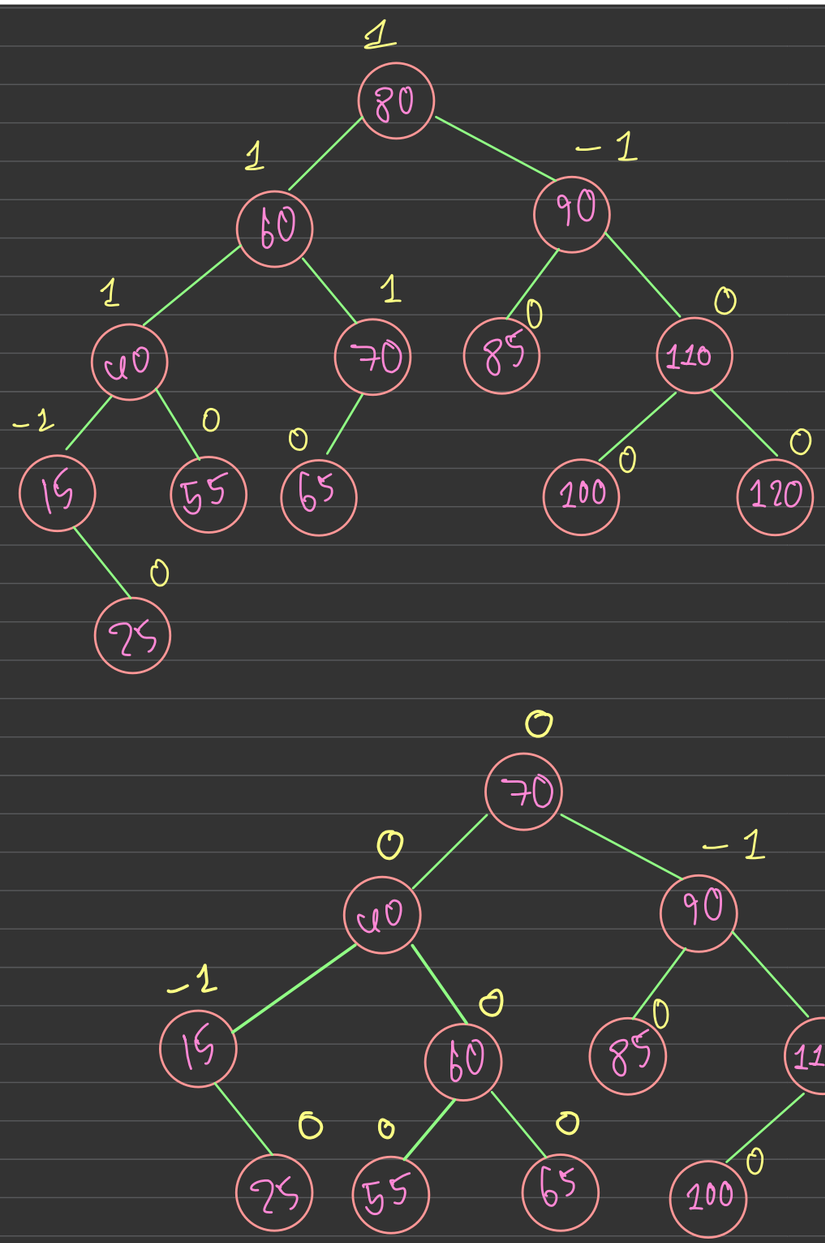

0
80
0
60
-1
90
0
40
1
70
0
85
1
120
0
15
0
55
0
65
0
100


1
80
1
60
-1
90
1
40
1
70
0
85
1
120
-1
15
0
55
0
65
0
100
0
25


0
80
1
60
-2
90
1
40
1
70
0
85
2
120
-1
15
0
55
0
65
-1
100
0
25
0
110

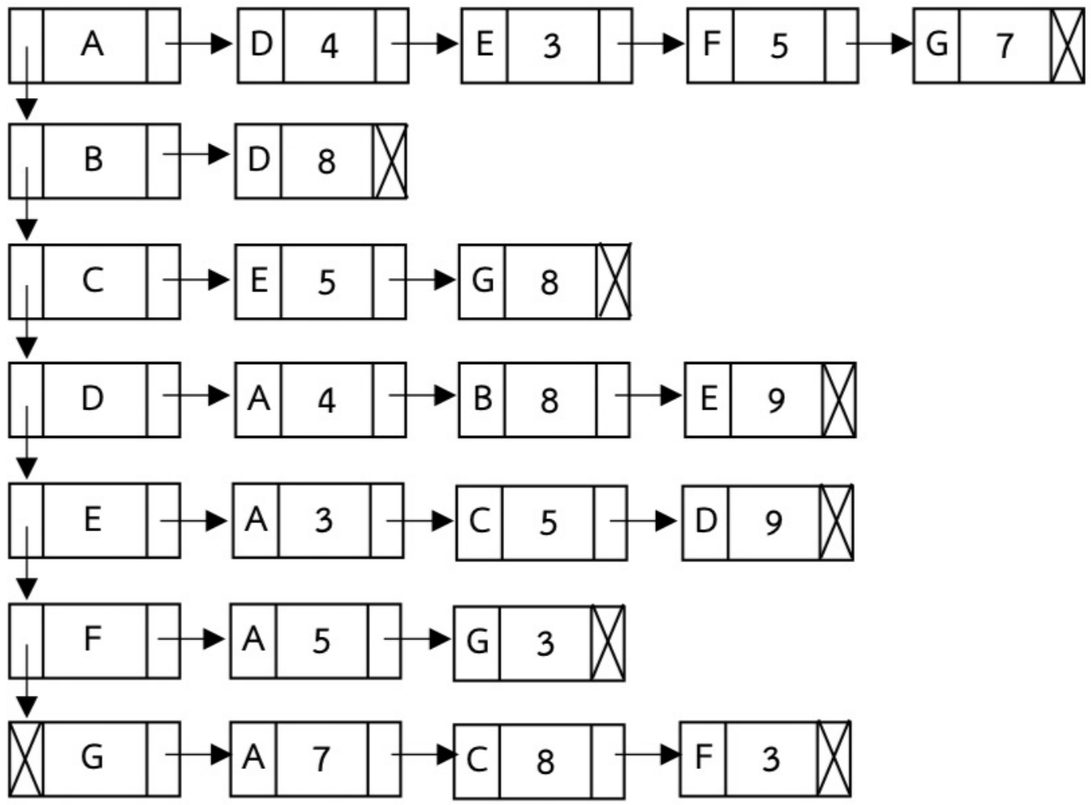

1
80
1
60
-1
90
1
40
1
70
0
85
0
110
-1
15
0
55
0
65
0
100
0
120
0
25

4. จงลบโหนด 80 ออกจาก AVL ผลลัพธ์ของข้อ 3 แล้วแสดง AVL ผลลัพธ์หลังจากการลบโหนด



บทที่ 6 : Graph

กำหนดให้ กราฟ G1 มี Adjacency List ดังนี้



1. จงวาดรูปกราฟ G1
2. จงเขียน Minimum Spanning Tree ของกราฟนี้
3. จงเขียน Shortest Path จาก Vertex G ไปยัง Vertex อื่นๆ