

Binary Search Tree

1. จงเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

0. **BST** tree;

1. tree.insert('H');

2. tree.insert('A');

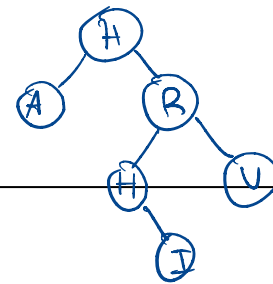
3. tree.insert('R');

4. tree.insert('H');

5. tree.insert('U');

6. tree.insert('I');

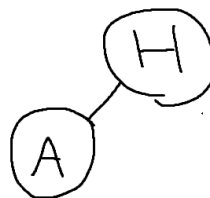
A B C D E F G H I J K
L M N O P Q R S T U V
W X Y Z



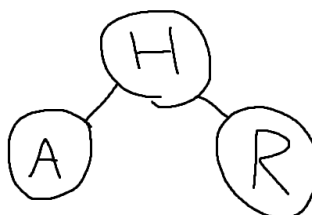
1.



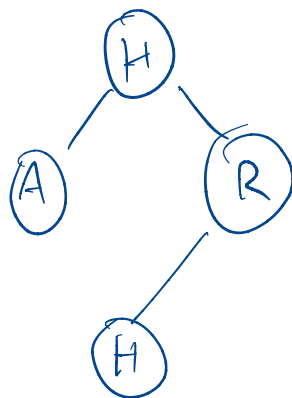
2.



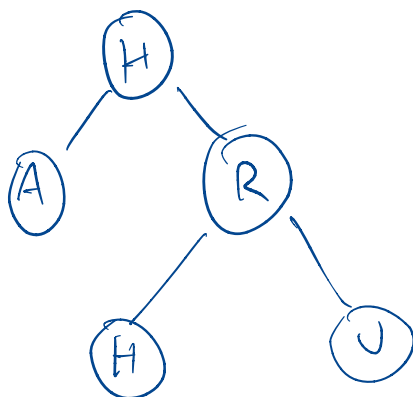
3.



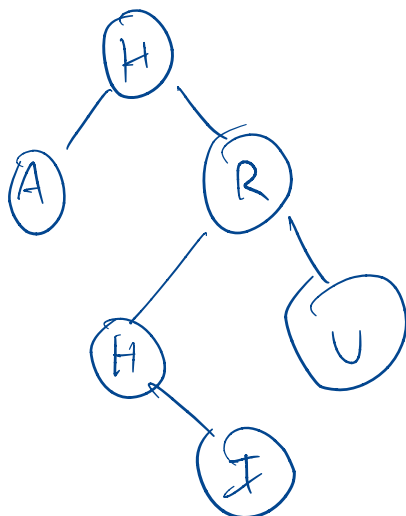
4.



5.



6.



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น H A R H I U

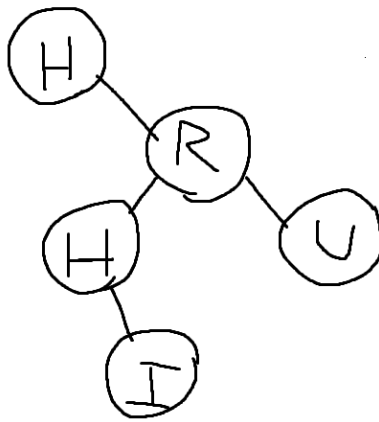
หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น A H H I R U

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น A I H U R H

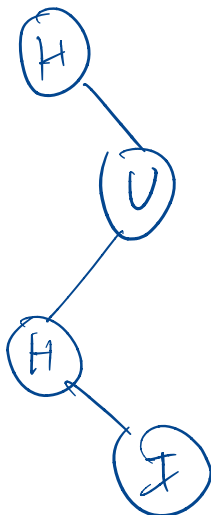
2. ต่อจากข้อ 1 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

```
7.delete_node(&(tree.root->left)); // A  
8.delete_node(&(tree.root->right));  
9.delete_node(&(tree.root->right));
```

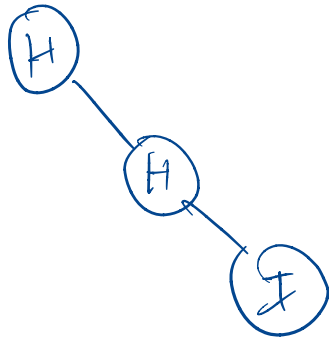
7.



8.



9.



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น H H I

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น H H I

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น I H H

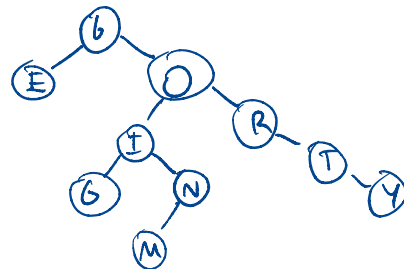
3. จงเขียนแผนภาพการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว (ออกแบบบรรทัดเองเลยครับ)

```

0.  BST tree2;
1.  tree2.insert('G');
2.  tree2.insert('O');
3.  tree2.insert('I');
4.  tree2.insert('N');
5.  tree2.insert('G');
6.  tree2.insert('M');
7.  tree2.insert('E');
8.  tree2.insert('R');
9.  tree2.insert('T');
10. tree2.insert('Y');

```

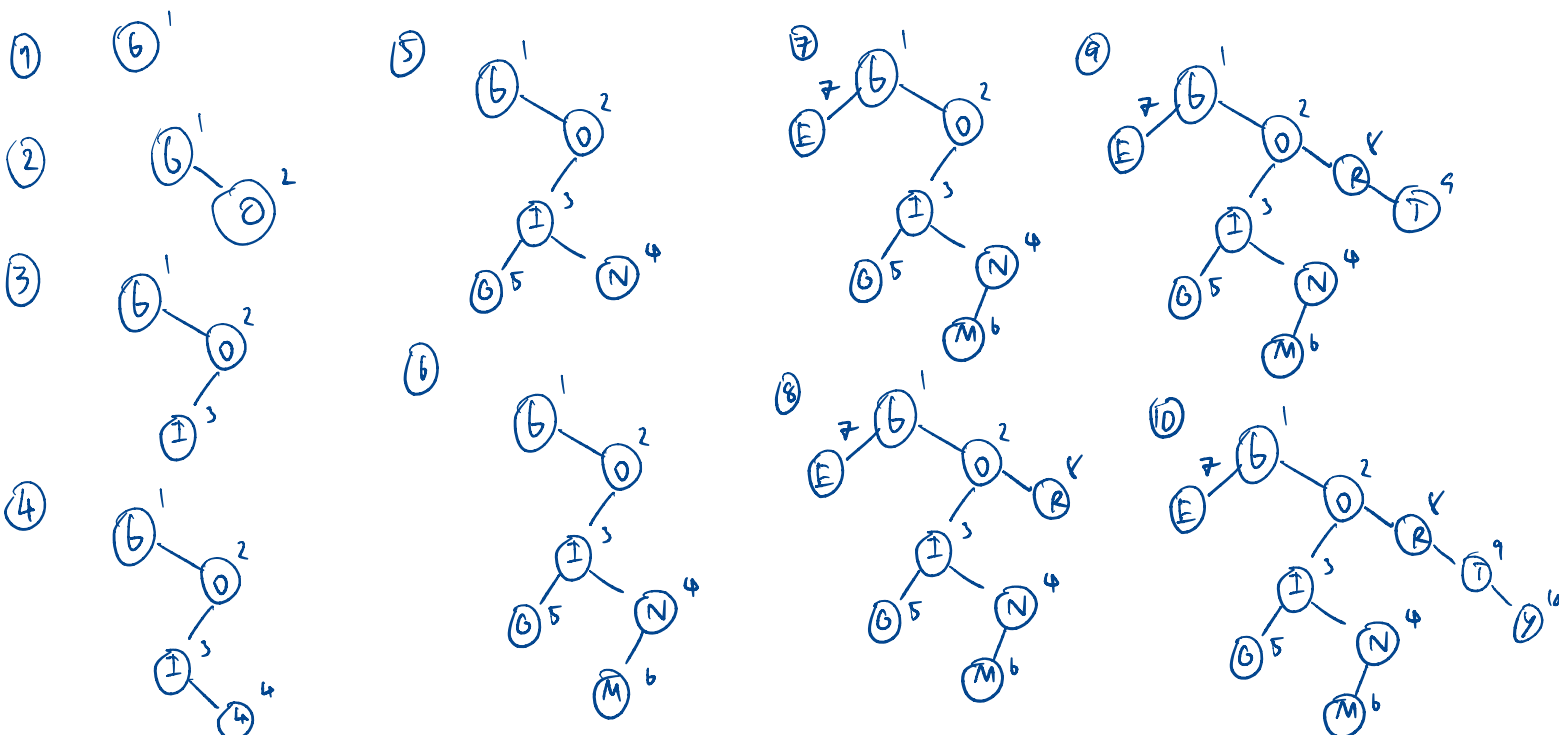
A B C D E F G H I
J K L M N O P Q R S
T U V W X Y Z



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น GEIOGNMRTY

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น E G G I M N O R T Y

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น E G M N I Y T R O G

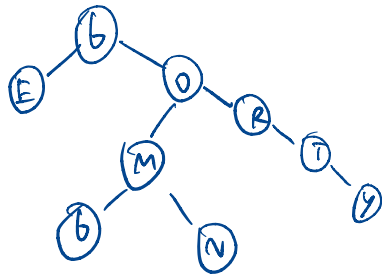


หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น
หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น
หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น

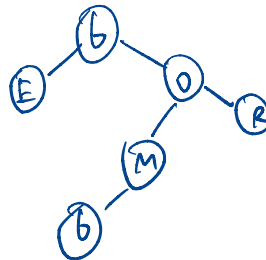
4. ต่อจากข้อ 3 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

```
11. delete_node(&(tree2.root->right->left));  
12. delete_node(&((tree2.root->right->left)->right));  
13. delete_node(&((tree2.root->right->right)->right));  
14. delete_node(&((tree2.root->right->right)->right));
```

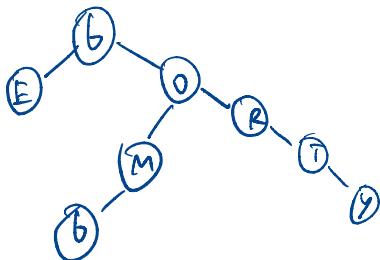
(1)



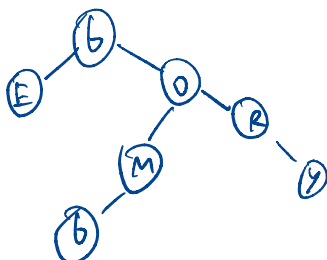
(4)



(2)



(3)



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น 6 E O M O R

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น E G G M O R

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น E G M R O G

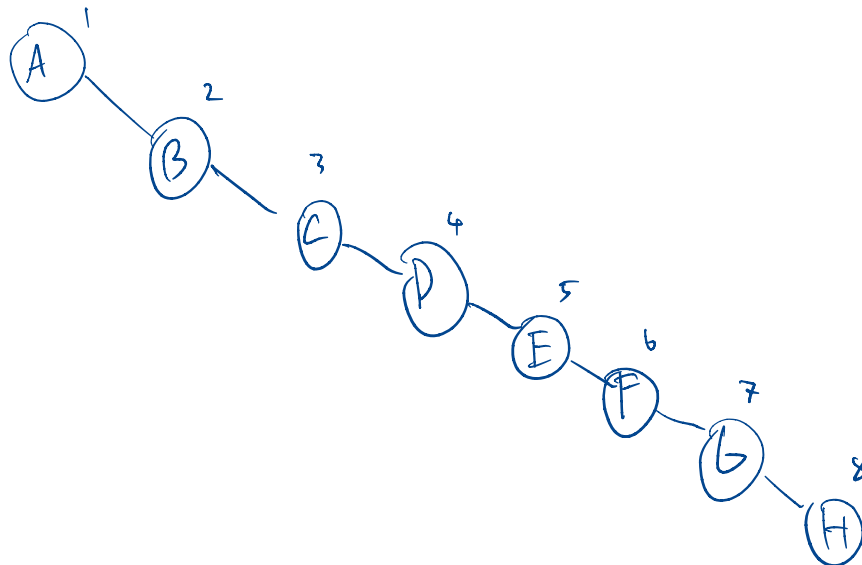
5. จงเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว (ออกแบบบรรทัดเองเลยครับ)

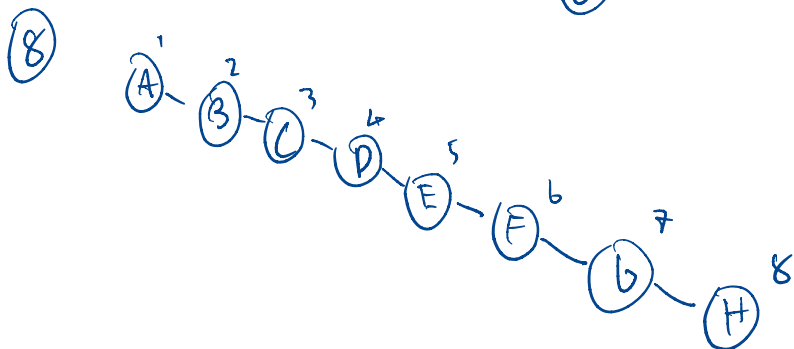
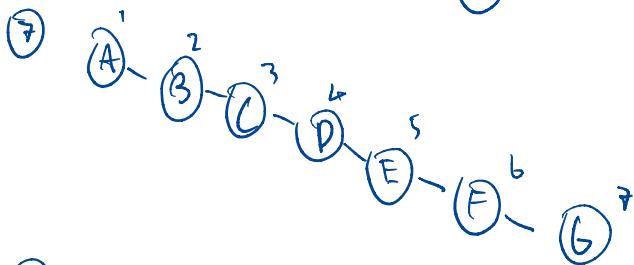
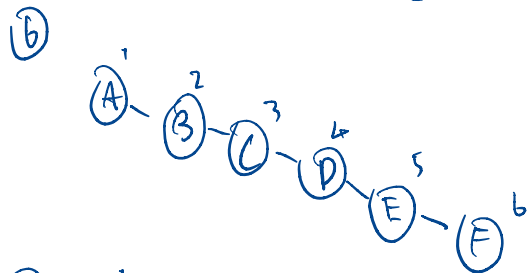
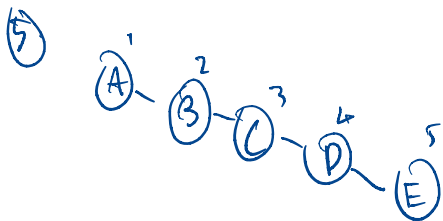
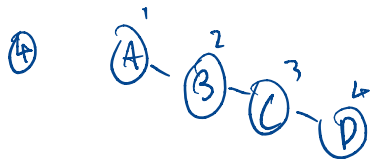
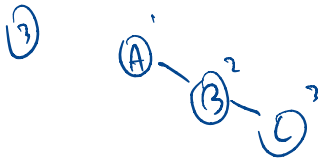
```
1.  BST tree3;  
2.  tree3.insert('A');  
3.  tree3.insert('B');  
4.  tree3.insert('C');  
5.  tree3.insert('D');  
6.  tree3.insert('E');  
7.  tree3.insert('F');  
8.  tree3.insert('G');  
9.  tree3.insert('H');
```

หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น A B C D E F G H

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น A B C D E F G H

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น H G F E D C B A

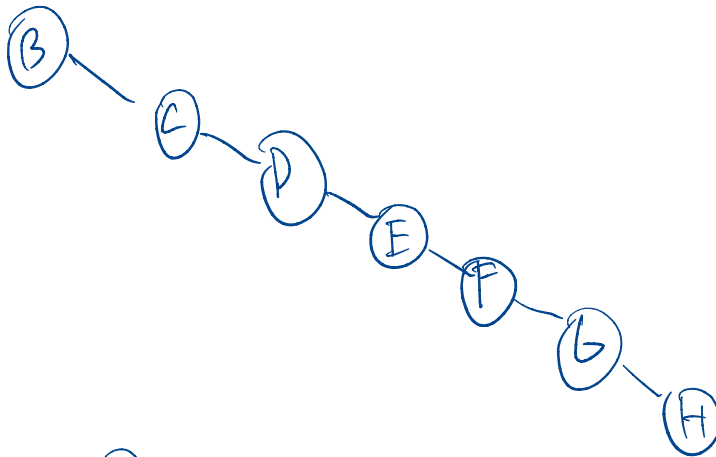




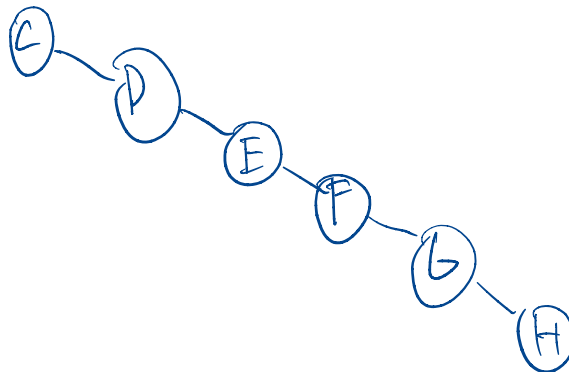
6. ต่อจากข้อ 3 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

```
10. delete_node(&(tree3.root));  
11. delete_node(&(tree3.root));  
12. delete_node(&(tree3.root));  
13. delete_node(&(tree3.root));
```

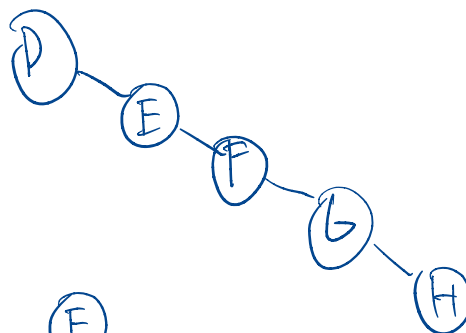
10



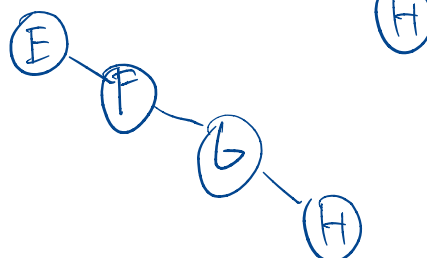
11



12



13



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น E F G H

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น E F G H

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น H G F E

7. BST ที่ balance กับ BST ที่ไม่ balance แบบไหนมีลำดับชั้นที่มากกว่ากัน หากจำนวนสมาชิกเท่ากัน เนื่องจากอะไร (ขอสั้นๆ)

ไม่ balance ลำดับชั้นมากกว่า เพราะจำนวนสมาชิกที่ balance จะรับซ้าย-ขวา แต่ถ้าไม่ balance จะกองกับอยู่ฝั่งเดียว ทำให้ชั้นมากกว่า

8. BST ที่ balance กับ BST ที่ไม่ balance หากต้องการ search แบบไหน ให้เวลาในการค้นหาน้อยกว่ากัน อย่างไร (ขอสั้นๆ)

balance 9 เวลาน้อยกว่า เพราะลำดับชั้นน้อยจะค้นหาข้อมูลน้อยกว่า จำนวนมีการค้นหาน้อยกว่าแล้ว

9. Tree ที่ balance กับ tree ที่ไม่ balance แบบใดโดยทั่วไปจะมีประสิทธิภาพดีไปกว่ากัน (ขอ1 คำ)

balance tree

10. ดังนั้นการคิด algorithm และ data structure เราควรพยายามให้ tree อยู่ในรูปของ balance หรือ unbalance เนื่องจากอะไร (ขอยาวๆ)

balance เพราะถ้า balance ข้อมูลจะถูกจัดให้เป็นระเบียบกว่า มองหาข้อมูลสะดวก ไม่ถึงจะเข้าถึงข้อมูล ก็ยังบอกถึงข้อมูลนั้นเจอได้ ถ้าแบบ ไม่ balance อาจต้องเข้าถึงข้อมูลทุกตัวถึงจะเจอตัวที่ต้องการ