

Binary Search Tree

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

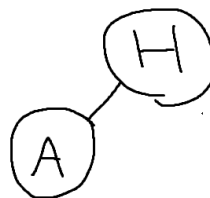
```
0.   BST tree;  
1.   tree.insert('H');  
2.   tree.insert('A');  
3.   tree.insert('R');  
4.   tree.insert('H');  
5.   tree.insert('U');  
6.   tree.insert('I');
```

ข้อควรระวัง : ถ้าตัวใน = root
ให้ไปทางขวา

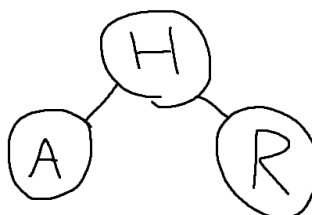
1.



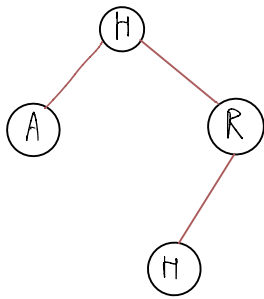
2.



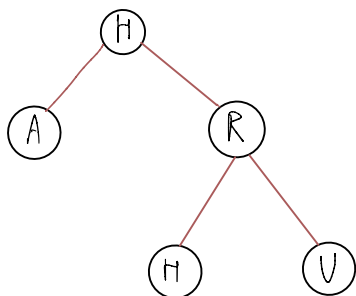
3.



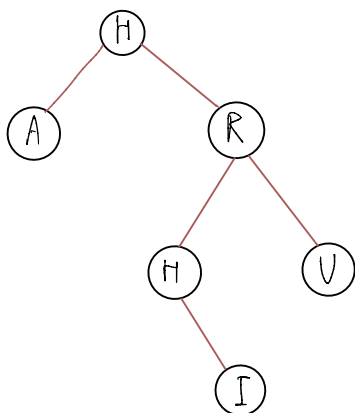
4.



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

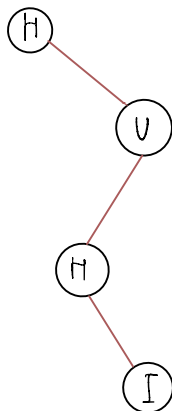
2. ต่อจากข้อ 1 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

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

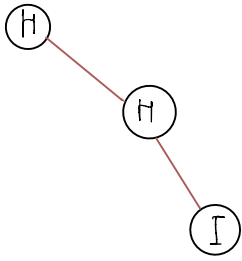
7.



8.



9.



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

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น HHI

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น IHH

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');
```

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

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น EGGIMNORTY

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น EGMNIYTROG

1)

G

2)

G

O

3)

G

O

I

4)

G

O

I

N

5)

G

O

I

G

N

6)

G

O

I

G

N

M

7)

G

E

O

I

G

N

M

8)

G

E

O

I

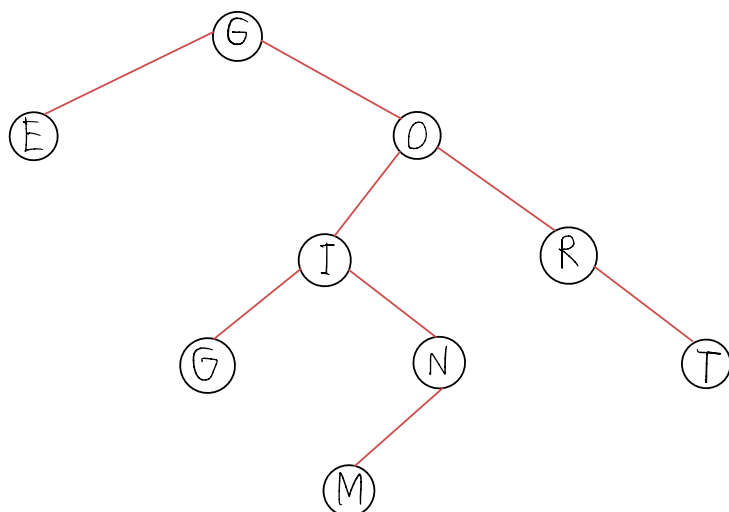
R

G

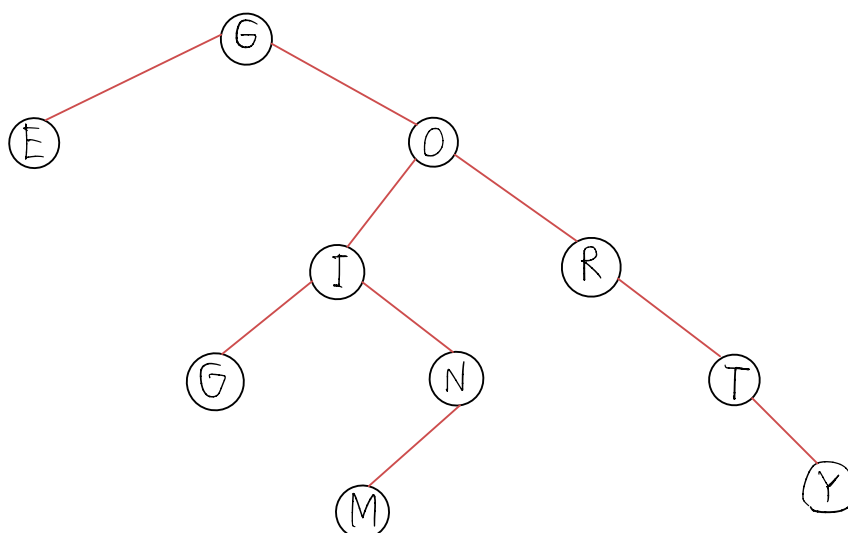
N

M

9)



10)



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

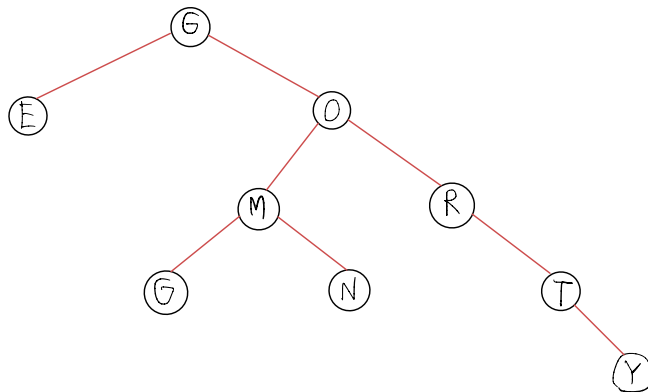
หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น EGGIMNORTY

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น EGMNIYTROG

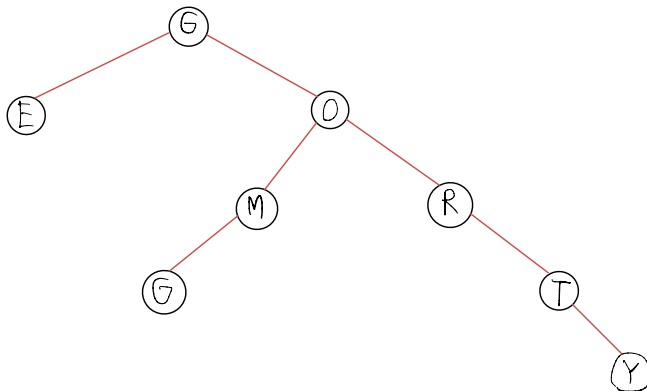
4. ต่อจากข้อ 3 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

```
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));
```

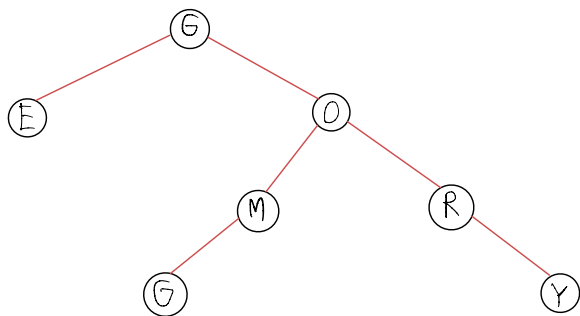
11.)



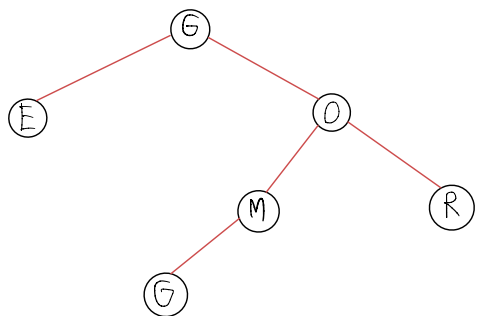
12.)



13)



14)



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น G E O M G 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 ABCDEFGH

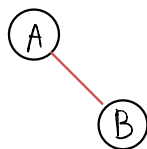
หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น ABCDEFGH

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น HGFEDCBA

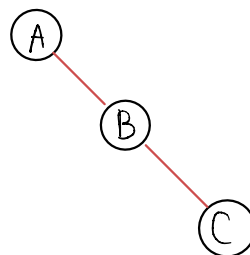
2)



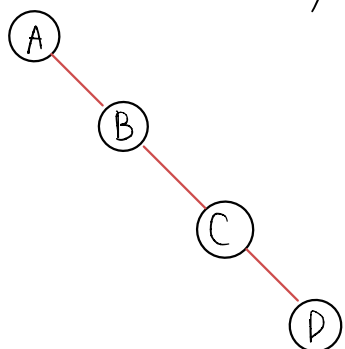
3)



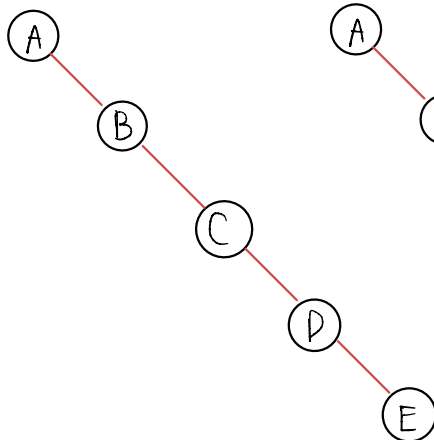
4)



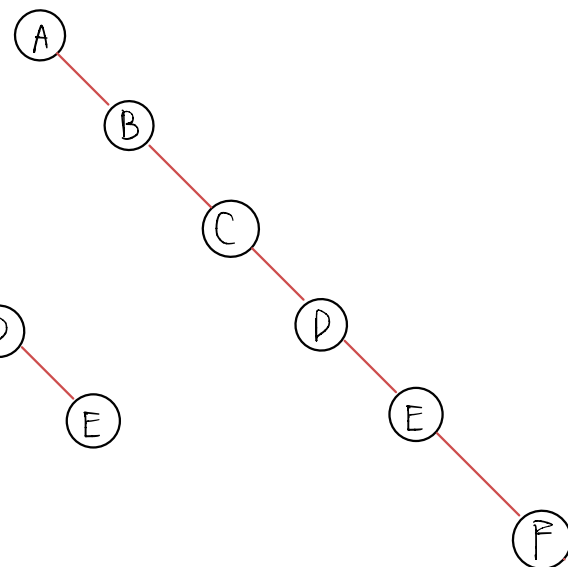
5)



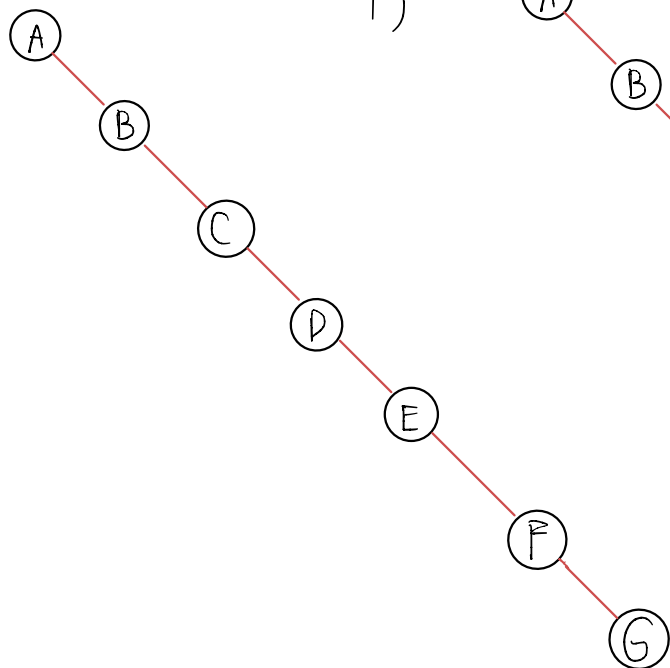
6)



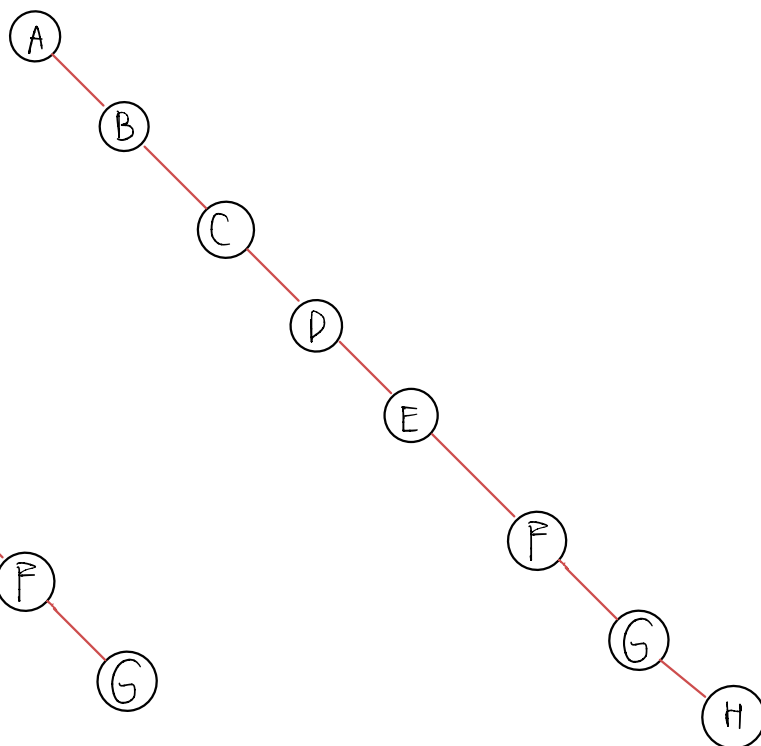
7)



8)

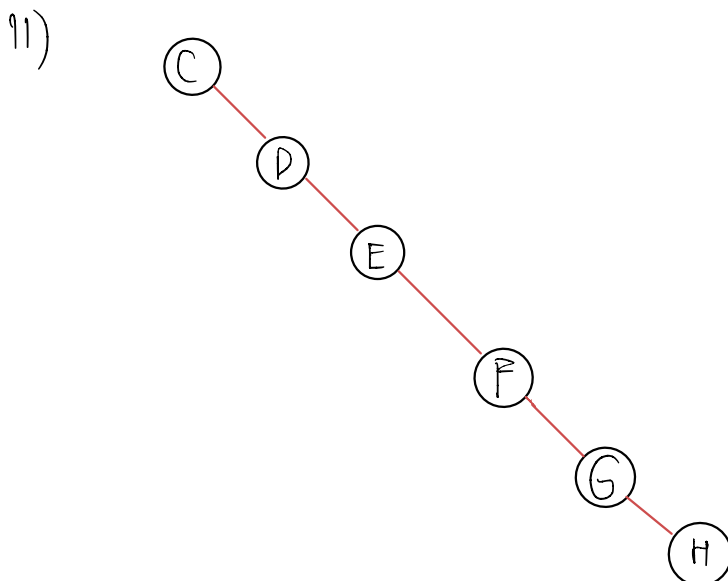
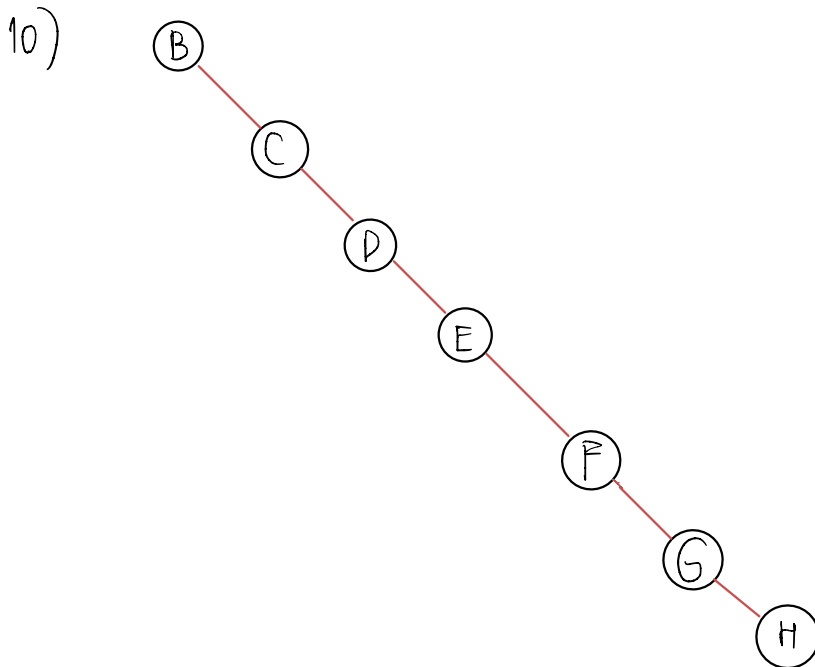


9)

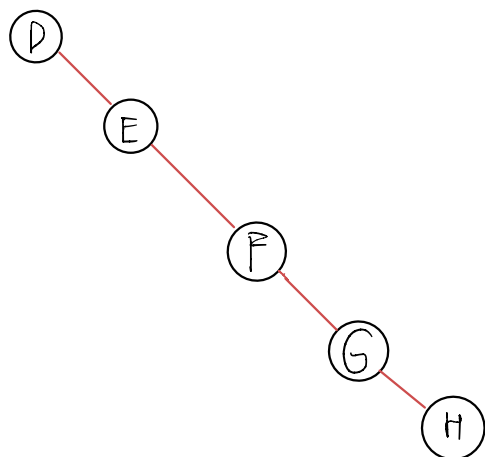


6. ต่อจากข้อ 3 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ละบรรทัด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

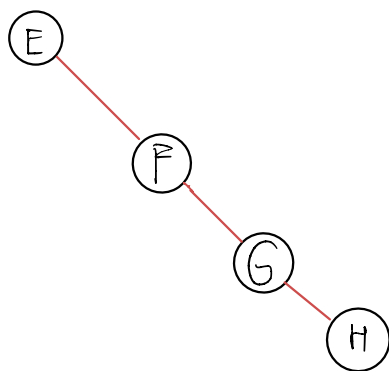
```
10. delete_node(&(tree3.root));  
11. delete_node(&(tree3.root));  
12. delete_node(&(tree3.root));  
13. delete_node(&(tree3.root));
```



12)



13)



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

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น EFGH

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น HGFE

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

.....แบบไม่ Balance เพราะ ข้อมูลมีการเรียงลำดับไม่ดี จึงหนักไปทางฝั่งใดฝั่งหนึ่ง

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

.....แบบ balance เพราะแบบ balance จะมีการเรียงลำดับแบบ in order อยู่แล้ว

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

.....balance

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

.....ควรพยายามให้ อยู่ในรูปแบบ balance เนื่องจากข้อมูลจะได้มีการจัดเรียง
อย่างเป็นระเบียบและเรียงลำดับ ซึ่งจะทำให้การ search ข้อมูล มีความรวดเร็ว
และมีประสิทธิภาพมากกว่าแบบ ไม่ balance