

Binary Search Tree

1. ຈงເຂົ້ານແຜນກາພຂອງການທຳການຂອງ Binary search tree ໃນໂປຣແກຣມຕ່ອໄປນີ້ທີ່ລະບຽບທັດ ແລະ ຕອບຄໍາຖາມເກື່ອງກັບການທົ່ວ (Traversal) ໄປໃນ tree ດັ່ງກ່າວ

```

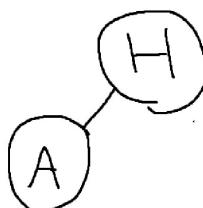
0.   BST tree;
1.   tree.insert('H');
2.   tree.insert('A');
3.   tree.insert('R');
4.   tree.insert('H');
5.   tree.insert('U');
6.   tree.insert('I');

```

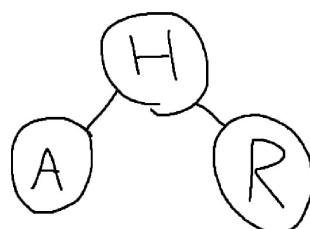
1.



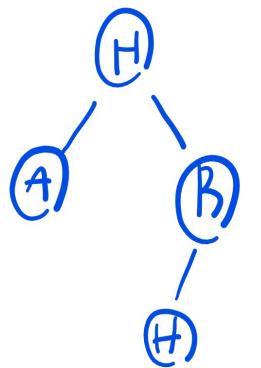
2.



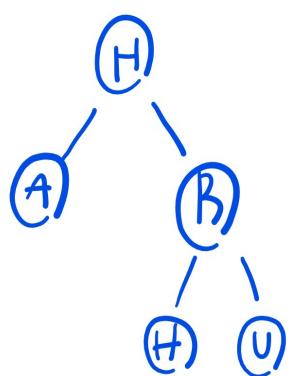
3.



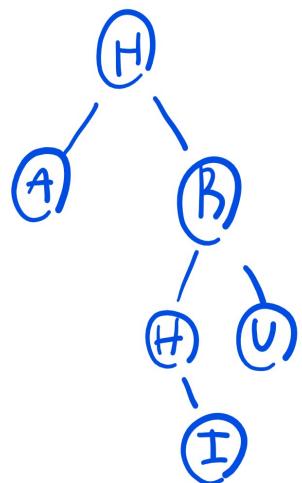
4.



5.



6.



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น **H A B H I U**.....

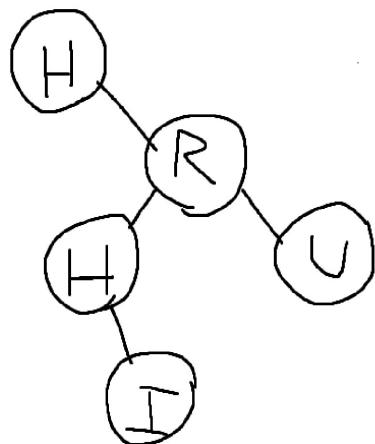
หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น A H H I R U

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

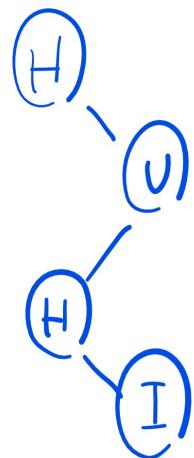
2. ต่อจากข้อ 1 หากใช้ code ดังต่อไปนี้ จงเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่จะบรรยาย และตอบคำถูกเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

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

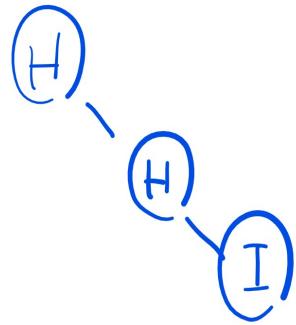
7.



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. จงเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ลະບຽບ และตอบ
คำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว (ออกแบบบรรทัดเองเลยครับ)

```

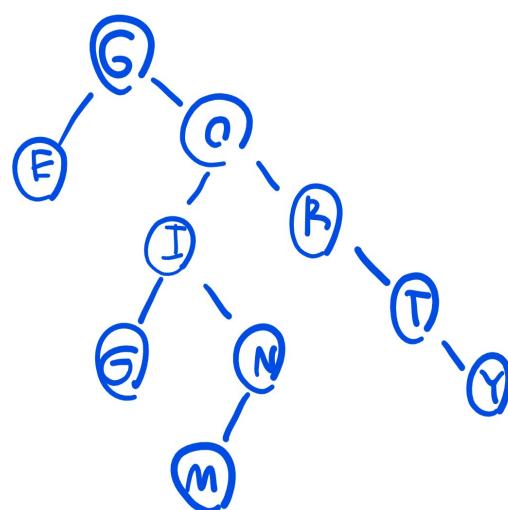
0. BST tree2;
1. tree2.insert('G');
2. tree2.insert('O');
3. tree2.insert('I');
4. tree2.insert('N');
5. tree2.insert('G');
6. tree2.insert('M');
7. tree2.insert('E');
8. tree2.insert('R');
9. tree2.insert('T');
10. tree2.insert('Y');

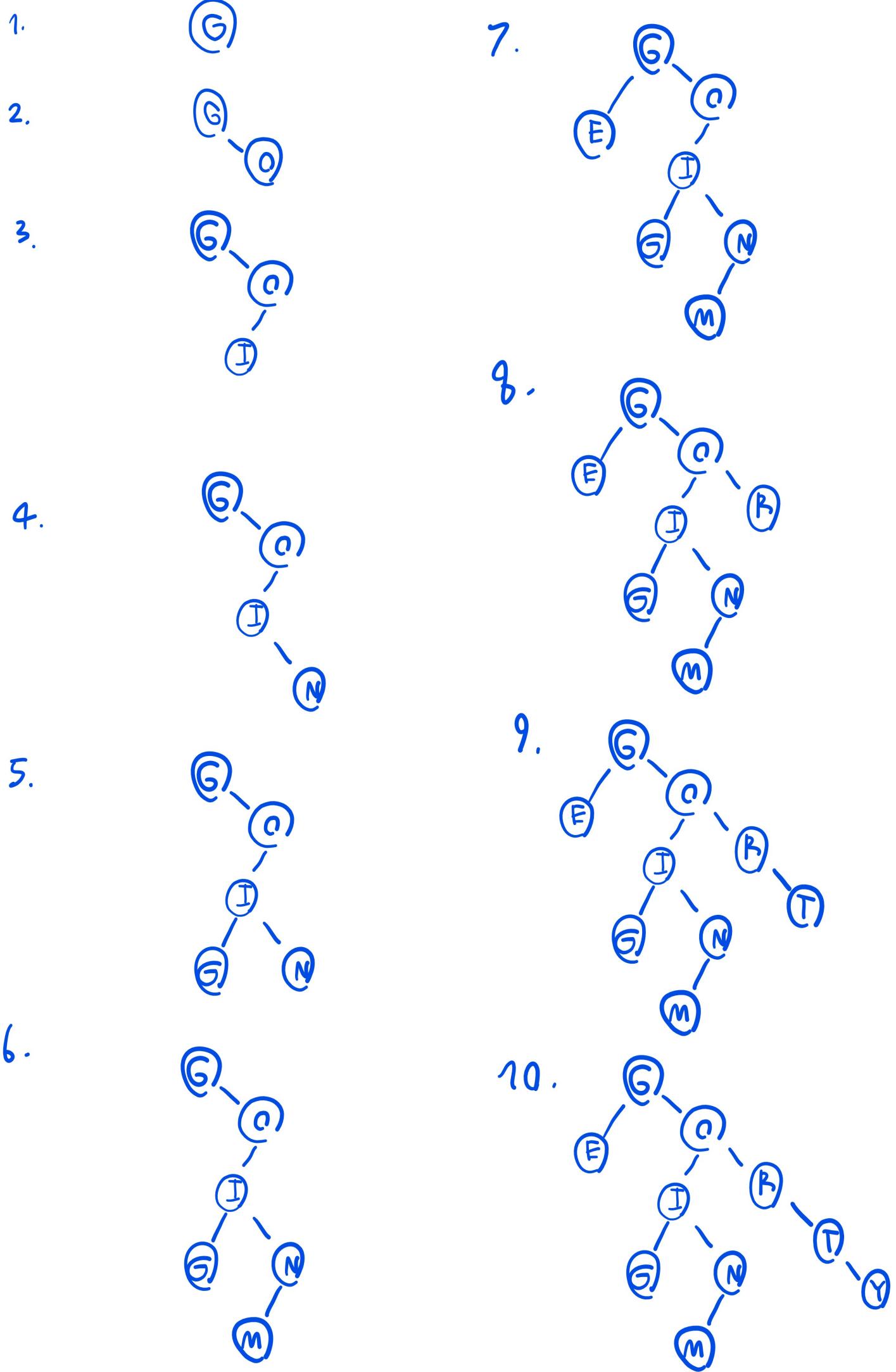
```

หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น **G E O I G N M B T Y**

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 B O G**

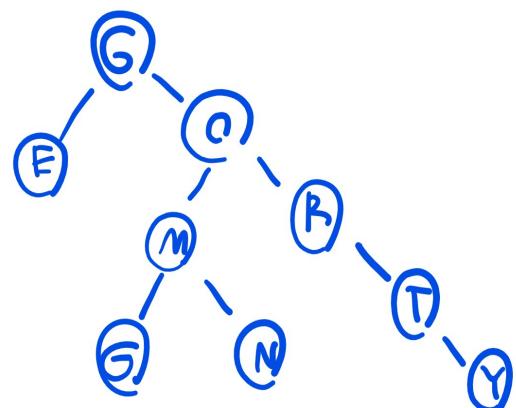
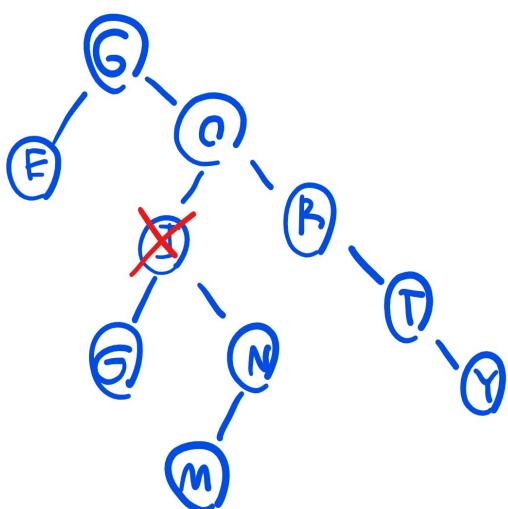




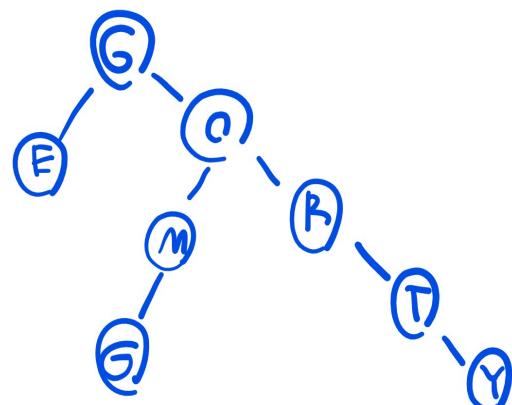
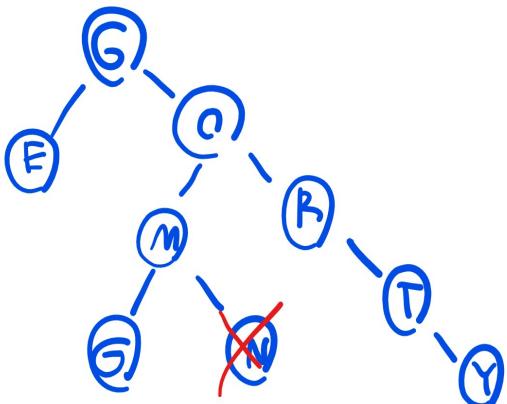
4. ต่อจากข้อ 3 หากใช้ code ดังต่อไปนี้ จะเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่จะบรรลุด และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

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));`

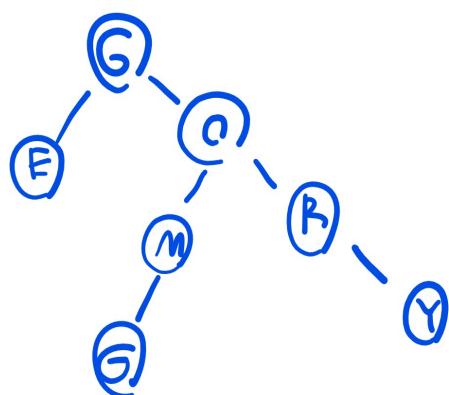
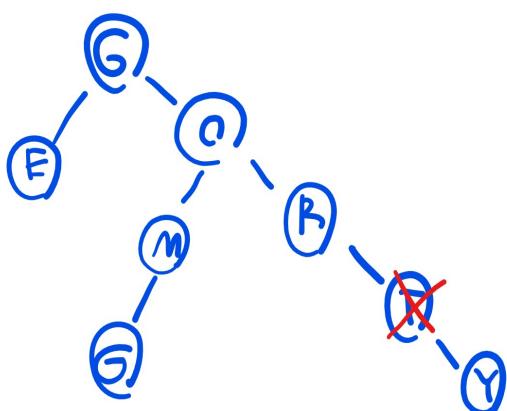
11.



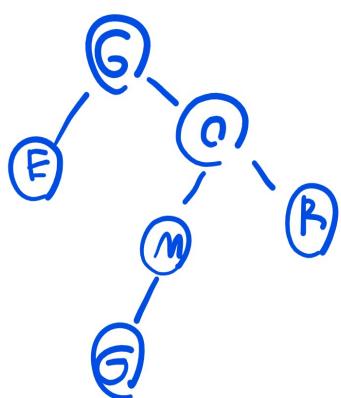
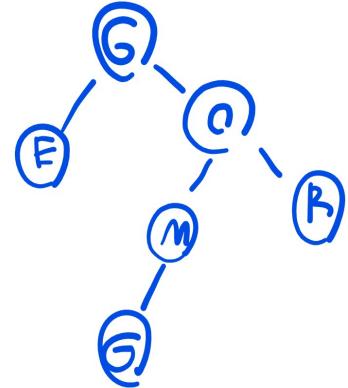
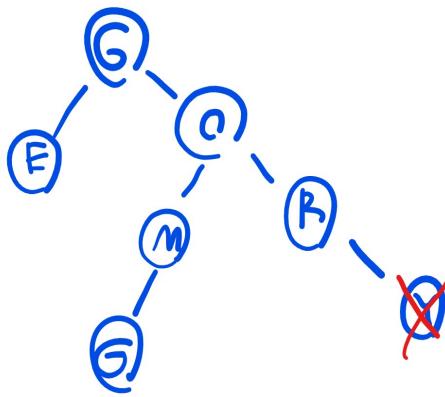
12.



13.



14.



หาก travers tree ดังกล่าว แบบ Pre-order จะได้ output เป็น

EGCMB

หาก travers tree ดังกล่าว แบบ In-order จะได้ output เป็น

EGBMC

หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น

EGMB06

5. จงเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่ลับบอร์ด และตอบ
คำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว (ออกแบบบรรทัดเองเลยครับ)

```

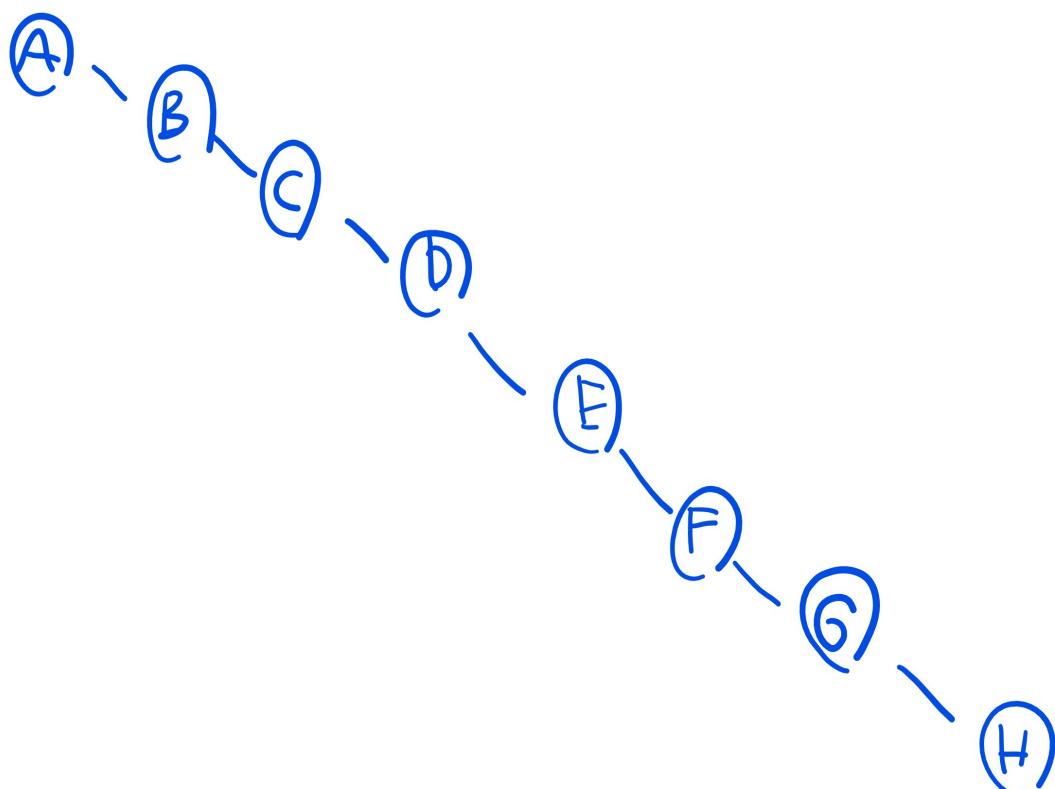
1. BST tree3;
2. tree3.insert('A');
3. tree3.insert('B');
4. tree3.insert('C');
5. tree3.insert('D');
6. tree3.insert('E');
7. tree3.insert('F');
8. tree3.insert('G');
9. tree3.insert('H');

```

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**

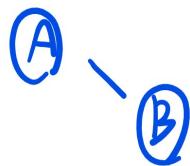
หาก travers tree ดังกล่าว แบบ Post-order จะได้ output เป็น **H G F E D C B A**



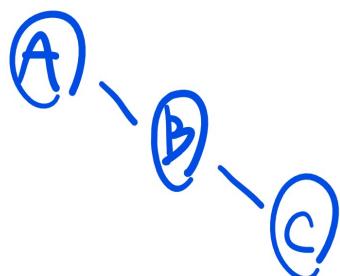
1.

(A)

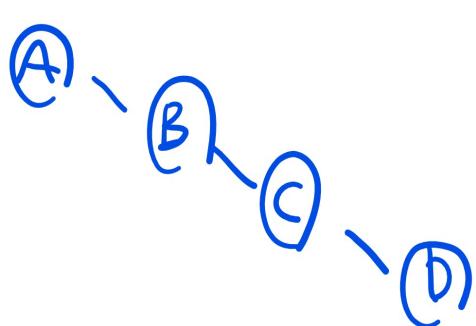
2.



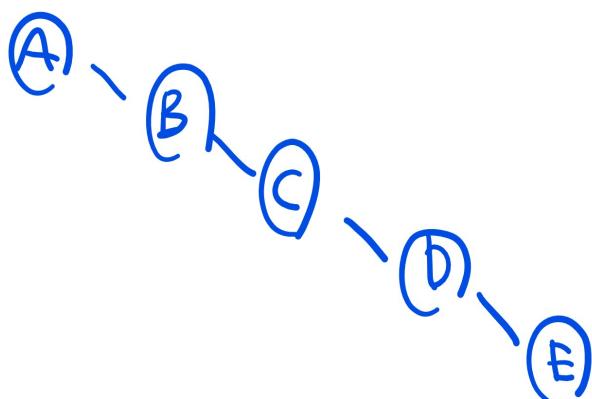
3.



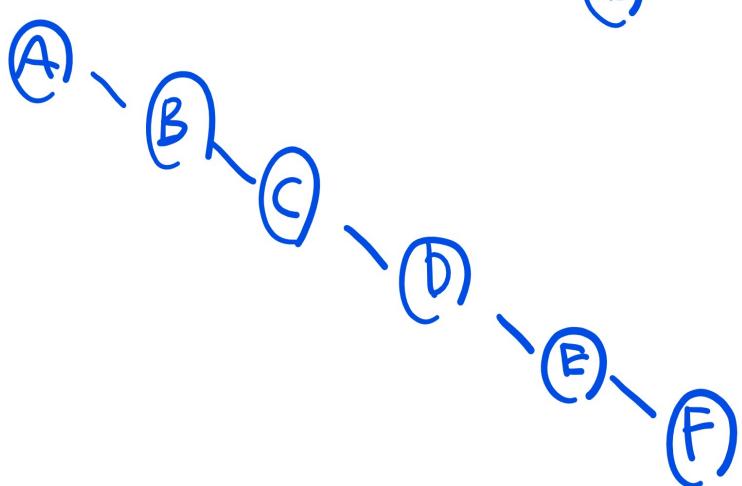
4.



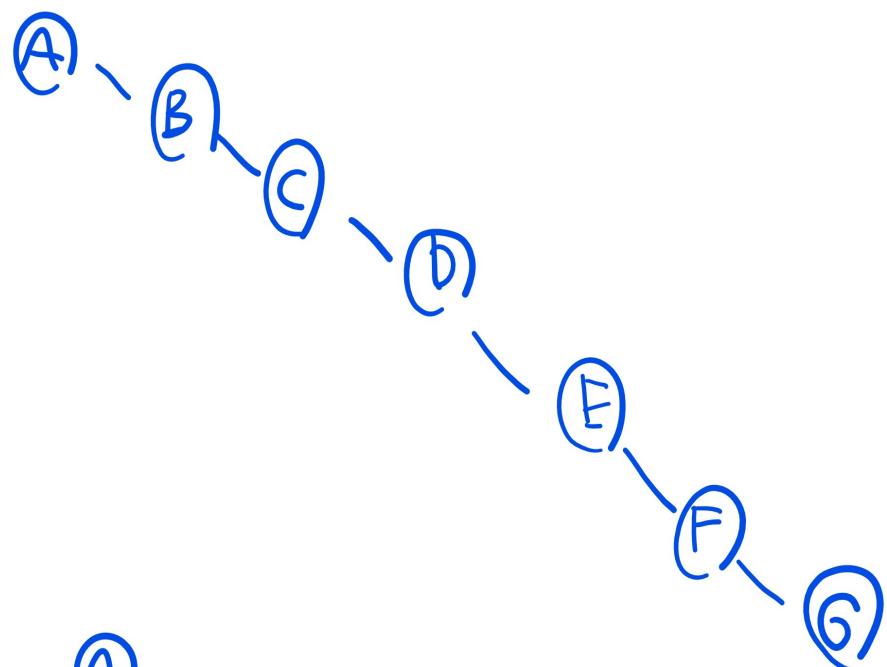
5.



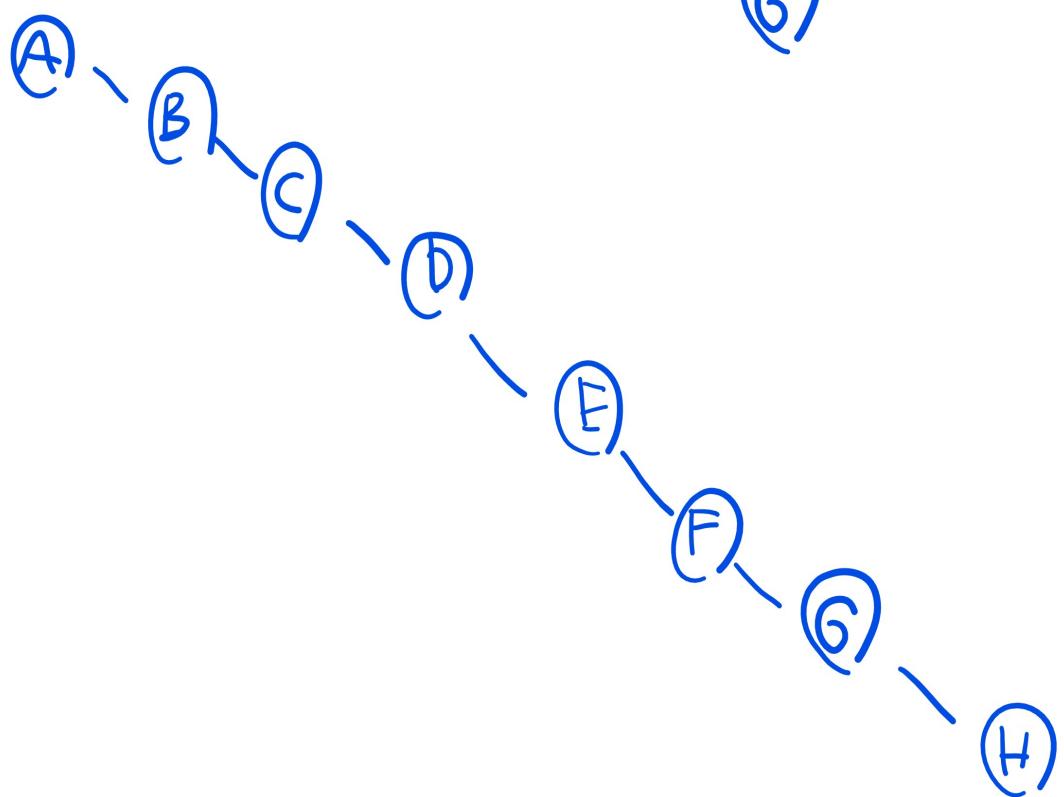
6.



7.



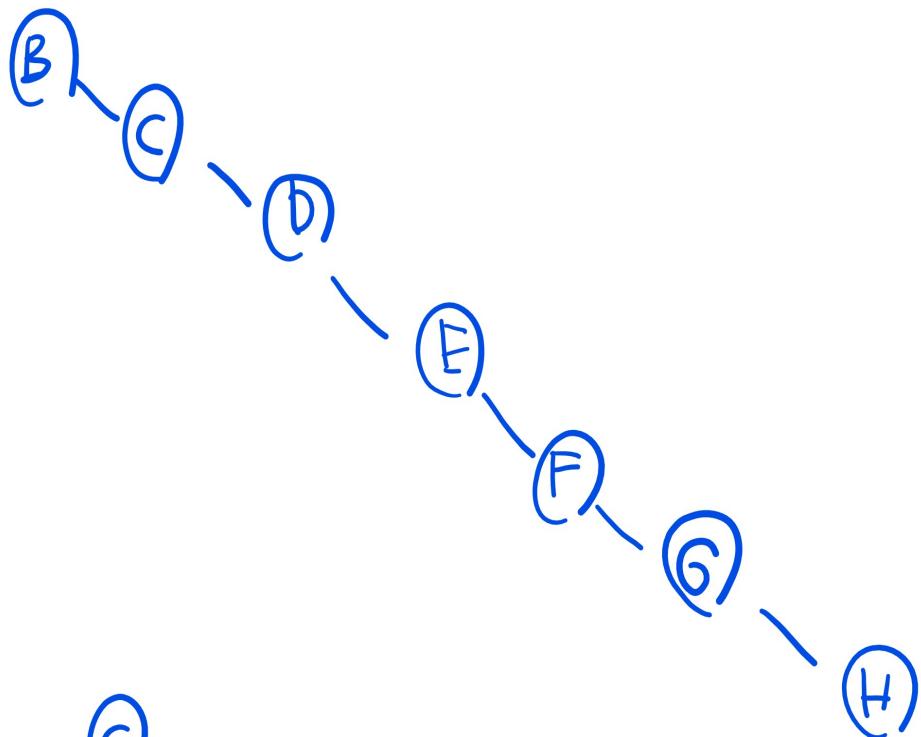
8.



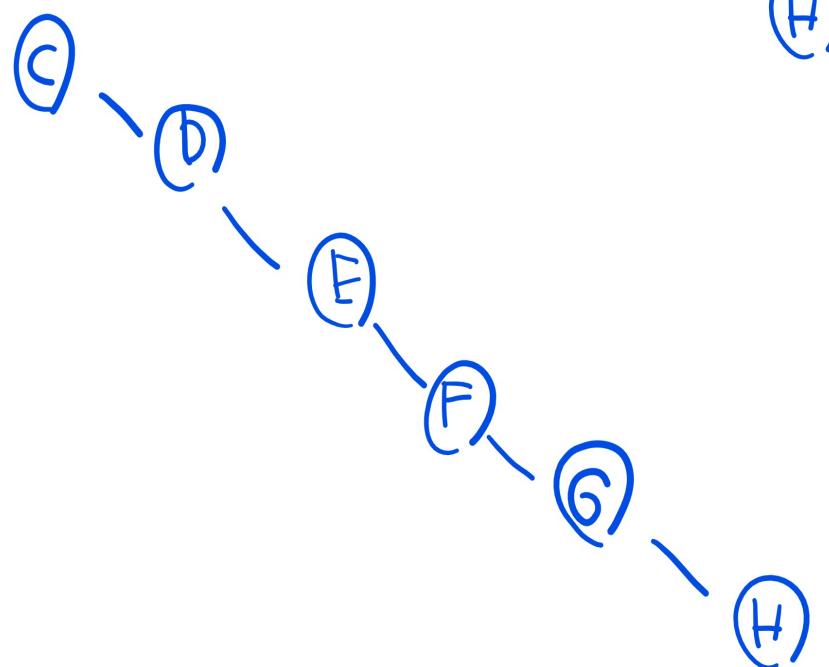
6. ต่อจากข้อ 3 หากใช้ code ดังต่อไปนี้ จะเขียนแผนภาพของการทำงานของ Binary search tree ในโปรแกรมต่อไปนี้ที่จะบรรยาย และตอบคำถามเกี่ยวกับการท่อง (Traversal) ไปใน tree ดังกล่าว

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

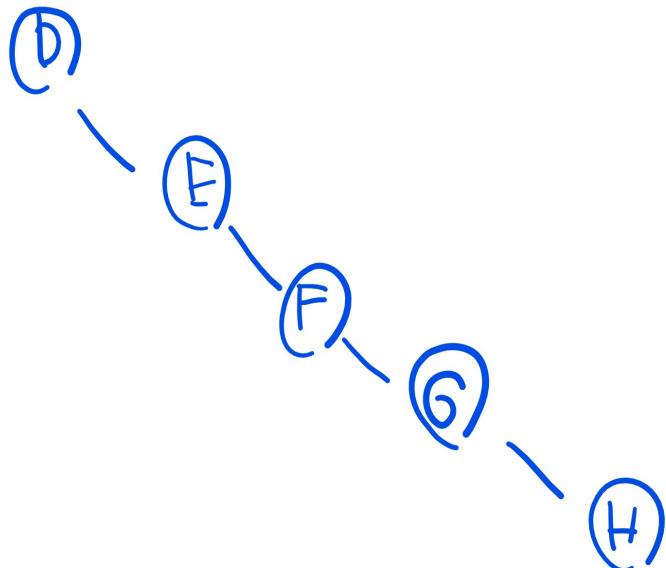
10.



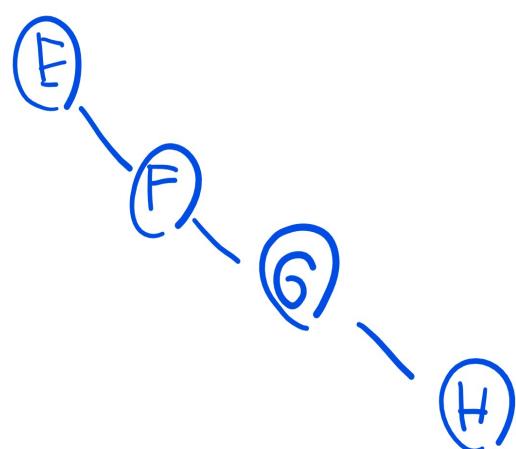
11.



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)

เมื่อ balance มีลำดับรากมากกว่า เพราะการที่จัดหาบุหรี่ให้กับ balance จะมีจำนวนสมาชิกที่เท่ากันทั้งสองฝ่ายและหากเป็นก่อให้เกิดตัวบุหรี่

8. BST ที่ balance กับ BST ที่ไม่ balance หากต้องการ search แบบไหน ให้เวลาในการค้นหาน้อยกว่ากัน

อย่างไร (ขอสั้นๆ)

เมื่อ balance ใช้วิธีการเดินทางขึ้นลงมากกว่า เพราะมีลำดับรากที่น้อยลง ทำให้ผู้คนสามารถเดินทางไปได้เร็วขึ้น

9. Tree ที่ balance กับ tree ที่ไม่ balance แบบใดโดยทั่วไปจะมีประสิทธิภาพดีกว่ากัน (ขอ 1 คำ)

balance tree

10. ดังนั้นการคิด algorithm และ data structure เราควรพยายามให้ tree อยู่ในรูปของ balance หรือ

unbalance เนื่องจากอะไร (ขอยาวๆ)

การที่เรา balance เพราะถ้าเป็นเมื่อ balance ขั้นตอนจะต้องดำเนินการนี้เป็นรูปแบบซ้ำๆ การหาขั้นตอนที่มีความซ้ำๆ ทำให้เราต้องใช้เวลาและพลังงานมาก แต่ถ้าเรา balance แล้วจะลดเวลาและพลังงานลง ทำให้เราสามารถทำงานได้เร็วขึ้น