BT-Ops

ให้เขียนโปรแกรมเพื่อสร้างและทำ Operation ต่าง ๆ ของ Binary Tree ที่มี Node structure ดังนี้

```
struct BinTree {
    int value;
    BinTree* left;
    BinTree* right;
};
```

โดยจะต้องทำ operation ตาม code ที่ได้จาก input ดังนี้

m = maximum หาค่าที่มากที่สุดใน Tree โดยการใช้ Tree Traversal และ Recursion

i = แสดง Output จาก In-order Traversal แบบที่แสดงค่า -1 แทน Missing Child ของ internal node ใด ๆ

a = แสดง Output จาก Pre-order Traversal แบบที่แสดงค่า -1 แทน Missing Child ของ internal node ใด ๆ

b = แสดง Output จาก Post-order Traversal แบบที่แสดงค่า -1 แทน Missing Child ของ internal node ใด ๆ

l = แสดง จำนวน Leave Nodes

p = แสดงค่าที่อยู่ใน Node Parent ของ Node ที่ระบุด้วยจำนวนเต็ม key ที่กำหนดค่าให้ หากไม่มี Node key ให้แสดง -1

- Input จะอยู่ในรูป p key คั่นด้วยอักขระว่าง ' '

s = แสดงค่าที่อยู่ใน Node Sibling ของ Node ที่ระบุด้วยจำนวนเต็ม key ที่กำหนดค่าให้ หากไม่มี Sibling หรือไม่มี Node key ให้แสดง -1

- Input จะอยู่ในรูป s key คั่นด้วยอักขระว่าง ' '

Input

- บรรทัดแรก คือจำนวนเต็มบวก h แทน Height ของ Binary Tree
- บรรทัดถัดมาจะเป็นจำนวนเต็ม 2^h - 1 จำนวนคั่นด้วยอักขระว่าง ' ' แทน Array Representation ของ Binary Tree ดังกล่าว โดยให้ Missing Children ของ internal Nodes แทนด้วย -1
- บรรทัดถัดมาคือ n แทนจำนวน Operation
- หลังจากนั้น n บรรทัด จะขึ้นต้นด้วยอักษรคำสั่ง m (max), i (in-order print), a (pre-order print), b (post-order print), l (leaves), p (parent search) หรือ s (sibling search)

Input

```
4
88 -1 32 -1 -1 7 26 -1 -1 -1 -1 -1 46 44 -1
11
m
i
a
b
l
s 88
p 32
s 26
p 44
s 99
p 99
```

Output

- แสดง output ตามที่ระบุด้วยชุดคำสั่งใน Input

output

```
88
-1 88 -1 7 46 32 44 26 -1
88 -1 32 7 -1 46 26 44 -1
-1 -1 46 7 44 -1 26 32 88
2
-1
88
7
26
-1
-1
```