# **Problem statements**

# HighOfNode

Tạo một cây nhị phân tìm kiếm T lưu trữ các số nguyên không trùng nhau (giá trị số nguyên < 1000, cây T có thể rỗng). Sau đó tính chiều cao h của một node ( biết h>=0 ).

## **INPUT**

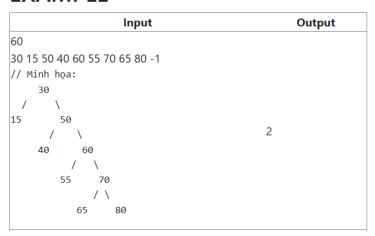
Số nguyên đầu tiên là node cần tìm chiều cao.

Số nguyên tiếp theo là gốc của cây. Các số nguyên còn lại sẽ được lần lượt thêm vào cây theo thứ tự nhập. Kết thúc khi gặp -1.

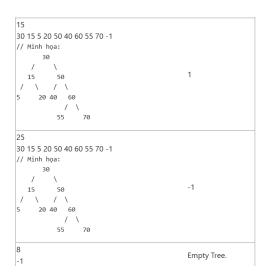
## **OUTPUT**

Xuất chiều cao h của node. Nếu node không có trong cây thì xuất "-1". Nếu cây rỗng thì xuất "Empty Tree.".

#### **EXAMPLE**



Problem statements 1



## CountNodeHave1Child

Tạo một cây nhị phân tìm kiếm T lưu trữ các số nguyên ( giá trị số nguyên < 1000, cây T có thể rỗng). Sau đó đếm xem trong cây có bao nhiều node có 1 cây con.

### **INPUT**

Số nguyên đầu tiên là gốc của cây. Các số nguyên sẽ được lần lượt thêm vào cây theo thứ tự nhập. Kết thúc khi gặp

## **OUTPUT**

Xuất số lượng node có 1 cây con trong cây.

### **FXAMPIF**

Problem statements 2

Input	Output
657281-1 // Minh họa: 6 / \ 5 7 / \ 2 8 /	3 // Node có 1 con: 5, 7, 2
4819-1 // Minh họa: 4 / \ 1 8 \ 9	1 // Node có 1 con: 8
6 -1	0
-1	0

# LNR\_BST(No\_stack\_recursive)

Tạo một cây nhị phân tìm kiếm T lưu trữ các số nguyên ( giá trị số nguyên < 1000, cây T có thể rỗng). Sau đó in cây theo thứ tự LNR. **Lưu ý dùng stack.** 

### **INPUT**

Số nguyên đầu tiên là gốc của cây. Các số nguyên sẽ được lần lượt thêm vào cây theo thứ tự nhập. Kết thúc khi gặp -1.

## **OUTPUT**

Xuất theo LNR. Các giá trị cách nhau một khoảng trắng. Nếu cây rỗng thì in ra "Empty Tree.".

### **EXAMPLE**

Input	Output
6 5 7 2 8 1 -1	1 2 5 6 7 8
4 8 1 9 -1	1 4 8 9
-1	Empty Tree.

Problem statements 3