Problem statements

Search in LL

Tạo danh sách liên kết đơn lưu trữ các số nguyên (giá trị số nguyên < 10,000). Các số được lần lượt thêm vào cuối danh sách, kết thúc khi gặp giá trị -1.
Hỏi trong danh sách có giá trị X hay không?

INPUT

Một dãy các con số nguyên trong đó:
- Các số nguyên liên tiếp sẽ được thêm vào danh sách bằng phương pháp thêm cuối, việc thêm vào kết thúc khi gặp -1.
- Số nguyên X sau số -1 là số cần tìm.

OUTPUT

In ra `true` nếu tìm thấy, in `false` nếu không tìm thấy.

EXAMPLE

Input	Output
4 8 2 5 1 8 -1 8	true
8 4 -1 9	false
-1 8	false

Insert in BST

Chèn giá trị X trong mảng 1 chiều lưu n phần tử (n <= 150,000) các số nguyên (< 1 tỷ) **TĂNG DẦN**. Sau khi chèn ta vẫn sẽ được mảng 1 chiều có thứ thự TĂNG DẦN. Lưu ý không dùng thuật toán sắp xếp.

Yêu cầu: Áp dung hàm tìm kiếm tuyến nhị phân **KHÔNG sử dụng đệ quy** .

INPUT

Dãy các số trong đó: (Giả sử luôn thỏa điều kiện nhập)

- Số nguyên đầu tiên: số nguyên X cần thêm vào.
- Số nguyên thứ hai: số lượng phần tử của mảng 1 chiều
- Các số nguyên còn lại: giá trị của các phần tử của mảng

OUTPUT

Xuất ra mảng số nguyên đã thêm X.

EXAMPLE

Input	Output
18 7 1 4 6 7 9 10 15	1 4 6 7 9 10 15 18
0 7 1 4 6 7 9 10 15	0 1 4 6 7 9 10 15
2 0	2

Count the number of nodes have 2 childs

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 nhiêu node có đủ 2 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 -1.

OUTPUT

Xuất số lượng node có đủ 2 cây con trong cây.

EXAMPLE

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

Sum all of node

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 tính tổng tất cả các node trong cây.

INPUT

Một dãy các số nguyên kết thúc là số -1.

OUTPUT

Xuất một số nguyên là tổng các node trong cây.

EXAMPLE

Input	Output
6 5 7 2 8 1 -1	29
4 8 1 9 -1	22
6 -1	6
-1	0

Join 2 Linked List

Viết hàm Join thực hiện nối 2 danh sách liên kết đơn L1 và L2 thành danh sách liên kết đơn L.
Biết rằng 2 danh sách liên kết đơn lưu trữ các số nguyên (giá trị số nguyên < 1000).
Sau đó thay tất cả các giá trị X trên L thành Y, việc thay đổi danh sách L này không ảnh hưởng danh sách L1, L2.
Ví dụ:
L1: 5 1 0 3 0
L2: 6 7 2 4
Sau khi nối:
L1: 5 1 0 3 0
L2: 6 7 2 4
L: 5 1 0 3 0 6 7 2 4
Sau đó đổi tất cả các giá trị 0 trong ds L thành 20 thì ta được:
L1: 5 1 0 3 9
L2: 6 7 2
L: 5 1 20 3 20 6 7 2 4

INPUT

3 dòng liên tiếp nhau với mỗi số nguyên trên mỗi dòng cách nhau 1 khoảng trống:
- Dòng 1: Các số lần lượt được thêm vào danh sách L1 bằng phương pháp thêm cuối, kết thúc việc tạo danh sách khi gặp số -1
- Dòng 2: Các số lần lượt được thêm vào danh sách L2 bằng phương pháp thêm cuối, kết thúc việc tạo danh sách khi gặp số -1
- Dòng 3: Số nguyên X, Y

OUTPUT

In ra 3 dòng liên tiếp với mỗi dòng gồm 1 dãy số cách nhau bởi một khoảng trắng tương ứng với 3 danh sách liên kết L, L1 và L2.

EXAMPLE

Input	Output
5 1 0 3 0 -1 6 7 2 4 -1 0 20	5 1 0 3 0 6 7 2 4 5 1 20 3 20 6 7 2 4
-1 -1 2 30	Empty List. Empty List. Empty List.

Add after prime number

Tạo một danh sách liên kết đơn L lưu trữ các số nguyên (giá trị số nguyên < 1000, danh sách L có thể rỗng). Sau đó thêm giá trị Y vào sau tất cả các số nguyên tố (nếu có) trong danh sách vừa tạo.

INPUT

Một dãy các con số nguyên trong đó:

- Các số nguyên liên tiếp sẽ được thêm vào danh sách bằng phương pháp `thêm cuối`, việc thêm vào kết thúc khi gặp -1.

- Số Y: số nguyên tiếp theo sau số -1.

OUTPUT

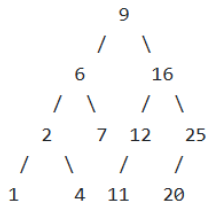
Danh sách sau khi thêm Y vào được in trên một dòng với mỗi số cách nhau một khoảng trắng.

EXAMPLE

Input	Output
6 5 7 2 8 1 -1 7	6 5 7 7 7 2 7 8 1
4 8 1 9 -1 10	4 8 1 9
-1 1	Empty List.

Level of Node

Tạo một cây nhị phân tìm kiếm T lưu trữ các số nguyên (giá trị số nguyên < 1000, cây T có thể rỗng).
Nhập vào một node và cho biết level của node đó (level >=0).
Ví dụ:



Nhập vào node 12. Level của node 12 là: 2.

INPUT

Số nguyên đầu tiên là node cần tìm level.

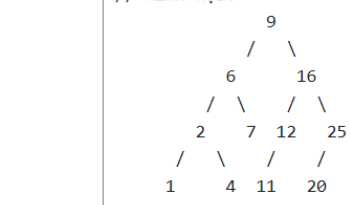
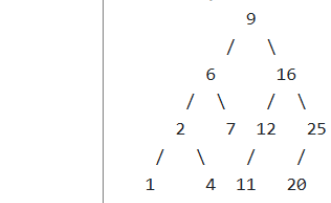
Số nguyên tiếp theo là gốc của cây. Các số nguyên còn lại sẽ được lần lượt thêm vào cây theo thứ tự nhập. Kết thúc khi gặp -1.

OUTPUT

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

EXAMPLE

Input	Output
7 9 16 6 7 2 12 4 11 25 1 20 -1 // Minh họa: 9 6 16 2 7 12 25 1 4 11 20	2
11 9 16 6 7 2 12 4 11 25 1 20 -1 // Minh họa: 9 6 16 2 7 12 25 1 4 11 20	3

11 9 16 6 7 2 12 4 11 25 1 20 -1 // Minh họa: 	3
30 9 16 6 7 2 12 4 11 25 1 20 -1 // Minh họa: 	-1
-1	Empty Tree.

IsMaxheap

Kiểm tra dãy số có phải dãy **maxheap** không?

INPUT

Dãy các số trong đó: (Giả sử luôn thỏa điều kiện nhập)

- Số nguyên đầu tiên: số lượng phần tử của mảng
- Các số nguyên còn lại: giá trị của các phần tử của mảng

OUTPUT

Nếu thỏa tính chất MaxHeap, xuất `true`. Ngược lại xuất `false`

EXAMPLE

Input	Output
10 10 9 5 5 6 4 0 1 2 4	true
0	true
2 8 9	false