Day 15: Linked List

Objective
Today we're working with *Linked Lists*. Check out the Tutorial tab for learning materials and an instructional video!

A *Node* class is provided for you in the editor. A *Node* object has an integer data field, $data$, and a *Node* instance pointer, $next$, pointing to another node (i.e.: the next node in a list).

A *Node insert* function is also declared in your editor. It has two parameters: a pointer, $head$, pointing to the first node of a linked list, and an integer $data$ value that must be added to the end of the list as a new *Node* object.

Task
Complete the *insert* function in your editor so that it creates a new *Node* (pass $data$ as the *Node* constructor argument) and inserts it at the tail of the linked list referenced by the $head$ parameter. Once the new node is added, return the reference to the $head$ node.

Note: If the $head$ argument passed to the *insert* function is *null*, then the initial list is empty.

Input Format

The *insert* function has 2 parameters: a pointer to a *Node* named $head$, and an integer value, $data$.
The constructor for *Node* has 1 parameter: an integer value for the $data$ field.

You *do not* need to read anything from stdin.

Output Format

Your *insert* function should return a reference to the $head$ node of the linked list.

Sample Input

The following input is handled for you by the locked code in the editor:
The first line contains *T*, the number of test cases.
The T subsequent lines of test cases each contain an integer to be inserted at the list's tail.

```
4
2
3
4
1
```

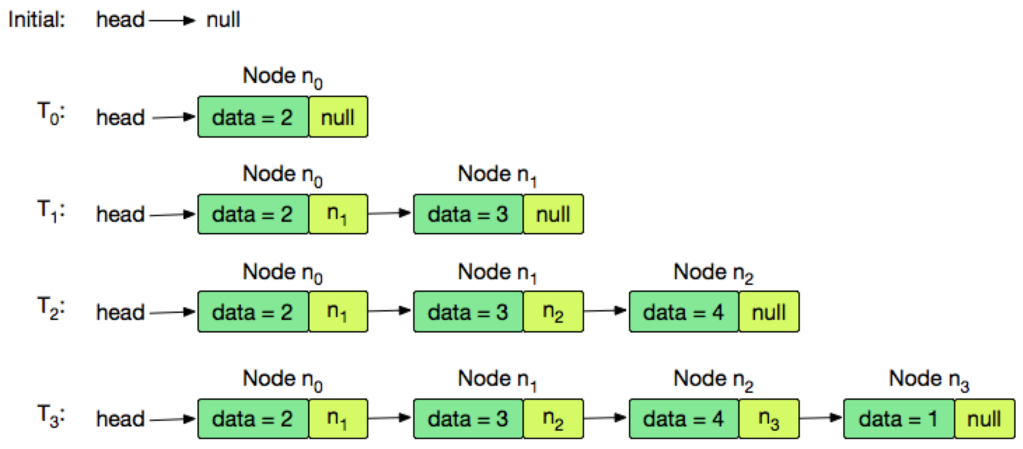
Sample Output

The locked code in your editor prints the ordered data values for each element in your list as a single line of space-separated integers:

```
2 3 4 1
```

Explanation

$T = 4$, so the locked code in the editor will be inserting 4 nodes.
The list is initially empty, so $head$ is null; accounting for this, our code returns a new node containing the data value 2 as the $head$ of our list. We then create and insert nodes 3, 4, and 1 at the tail of our list. The resulting list returned by the last call to $insert$ is $[2, 3, 4, 1]$, so the printed output is `2 3 4 1`.


Initial:
head
null
T0:
T1:
T2:
T3:
Node n0
Node n1
Node n2
Node n3
data = 2
data = 3
data = 4
data = 1
n1
n2
n3
null