left View:

We want to traverse the tree and print the left most node at each level.

Let i be a level in the tree that we just entered entered
We need to make sure that:
(*) we know if we already printed some node in level i
(**) we printed the left most node in level i

We can accomplish (**) by giving advantage to the left subtree, therefore we'll make the first recursive call to the left subtree.

We can accomplish (*) by maintaining two variables max_level and curr_level.

curr_level is the current level in which we are.

we define max_level by:
max_level = $\{ n \in \mathbb{N} \cup \{0\} \mid$ for any level i of G:
if $i \leq n$ then
we already printed a node
in level i :
if $n < i$ then we
didn't print any node on level i $\}$

Assume curr_level = 0
max_level = -1

```
leftView(r, curr_level, max_level): /* G's root = r */
1. if r = NIL then return max_level
2. if max_level < curr_level then do: /*we first entered to
                                        level curr_level */
   2.1. print key[r]
   2.2. max_level ← curr_level  /*update max_level */
3. max_level ← leftView(left[r], curr_level + 1, max_level)
4. max_level ← leftView(right[r], curr_level + 1, max_level)
5. return max_level
```

/* after 3 max_level may be updated so we get the updated value and use it in the function parameters at line 4.
at line 5 we're making sure to return the updated value of max_level */