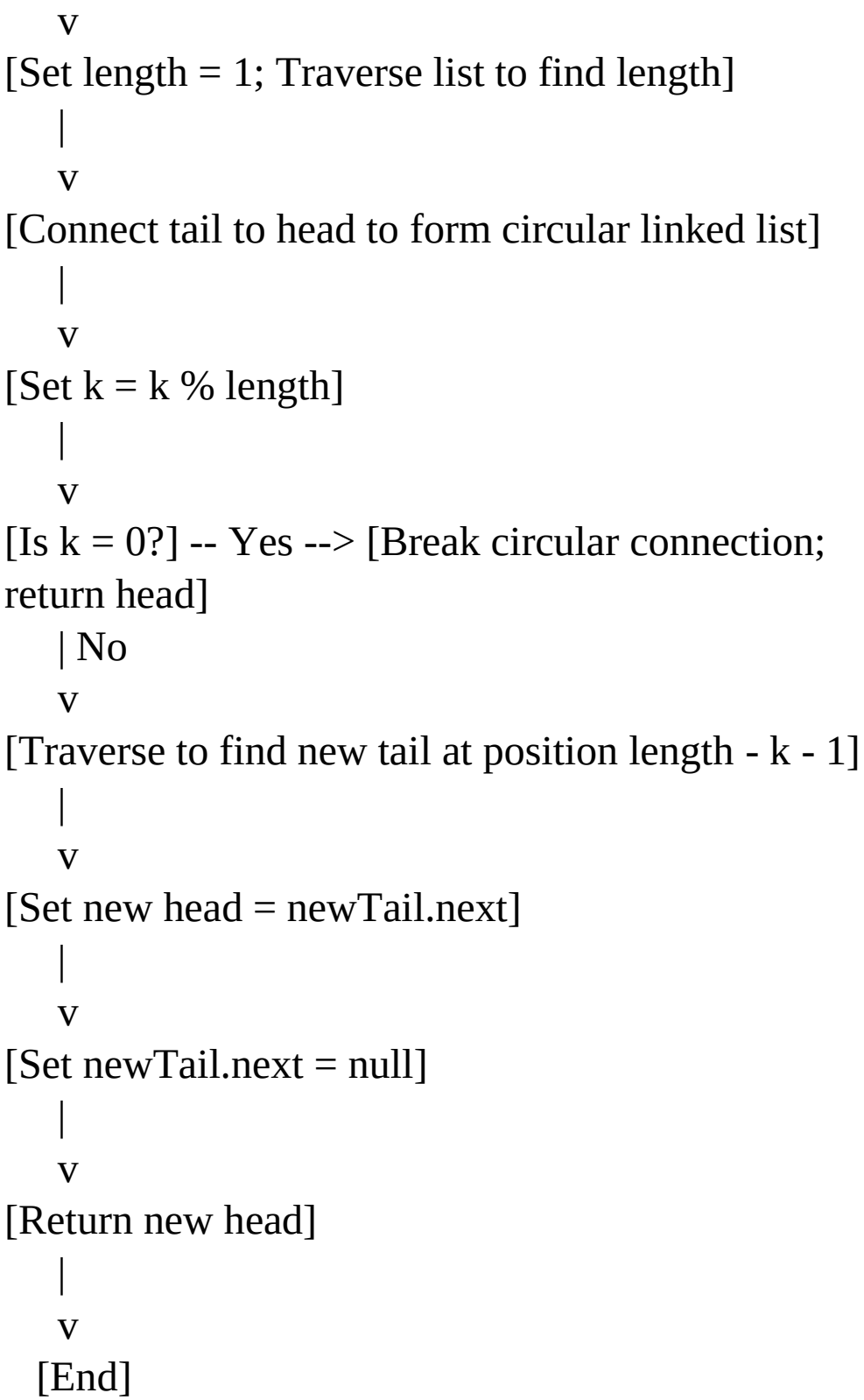
❖ Rotate List:

- Explanation:
  The program rotates a linked list to the right by k places, where each node in the list represents a digit. It first determines the length of the list, then adjusts the value of k to prevent unnecessary rotations. Finally, it rearranges the nodes by updating the pointers to form the new rotated list and returns the new head.
- Time Complexity and Space Complexity:
  - O(n) O(1)
- Flowchart:
  [Start]
  |
  v
  [Input number of nodes n]
  |
  v
  [Input values for the linked list]
  |
  v
  [Create linked list from input values]
  |
  v
  [Input number of places to rotate k]
  |
  v
  [Is head null OR head.next null OR k = 0?] -- Yes --> [Return head]
  | No


[Set length = 1; Traverse list to find length]
[Connect tail to head to form circular linked list]
[Set k = k % length]
[Is k = 0?] -- Yes --> [Break circular connection; return head]
| No
[Traverse to find new tail at position length - k - 1]
[Set new head = newTail.next]
[Set newTail.next = null]
[Return new head]
[End]