Algorithm: insertList(head, data, position)

Input: head - the head of the linked list

data - the data to be inserted into a new node

position - the position in the linked list where the new node should be inserted

Output: The head of the updated linked list

1. Initialize a counter 'count' to 0, a pointer 'prev' to null, a pointer 'current' to the head, and create a new node 'newNode' with the given data.

2. Check if the desired position is at the beginning (position == 0).

a. If true,

i. Set newNode.next to head.

ii. Return newNode as the new head of the linked list.

3. Traverse the linked list until reaching the desired position or the end of the list.

a. In each iteration,

i. Update prev to current.

ii. Move current to the next node.

iii. Increment count.

4. After the loop, if prev is not null,

a. Connect prev.next to newNode.

b. Connect newNode.next to current.

5. Return the head of the updated linked list.

Example usage:

- Create a linked list with a node containing data 1.

- Call insertList with the head, data 2, and position 1 to insert a new node at position 1.

- Call insertList with the head, data 3, and position 2 to insert a new node at position 2.

- The resulting linked list is 1 -> 2 -> 3.