```c
#include "linked-list.h"
#include "std-macros.h"

/****
 *
 * Implementation of linked-list.h.
 *
 */

LinkedList* newLinkedList() {
    LinkedList* list = new(LinkedList);
    list->head = null;
    list->length = null;
}

void insert(LinkedList* list, ListNode* node, int i) {

    ListNode* splice_node;              /* pointer to splice-in position */

    /*
     * Do nothing if i is out of range.
     */
    if (i < 0 || i > list->length) {
        return;
    }

    /*
     * If the list is empty, put the element at the head.
     */
    if (list->length == 0) {
        list->head = node;
    }

    /*
     * If i = 0, splice the node in at the head.
     */
    else if (i == 0) {
        node->next = list->head;
        list->head = node;
    }

    /*
     * Otherwise, splice the node in before the given position.
     */
    splice_node = getIthNode(list, i-1);
    node->next = splice_node->next;
    splice_node->next = node;
}

ListNode* getIthNode(LinkedList* list, int i) {

    ListNode* node = null;      /* Return value */
    int j;                      /* Search index */

    /*
     * Outta here if list is empty, i<0, or i>=list->length.
     */
    if (list->length == 0 || i < 0 || i >= list->length) {
        return null;
    }

    /*
     * Traverse the list with a for loop.  Note that there's nothing to do in
     * the loop body, since the bounds checks have already been taken care of.
     */
    for (node = list->head, j = 0; j < i; j++) ;

    /*
     * Return the located node.
     */
    return node;

}

void printList(LinkedList* list) {

    ListNode* node;                             /* traversal pointer */

    /*
     * Traverse the list, printing a comma after all but the last element.
     */
    for (node = list->head; node; node = node->next) {
        printf("%d%s", node->value, node->next ? "," : "");
    }

}
```