Linked Lists

---

1. Implement the removeNode() function for a linked list, using the lecture diagrams and pseudo-code as a reference. The prototype for the removeNode() function is given below:

```
int removeNode(ListNode **ptrHead, int index);
```

The function should return 0 if the delete operation is successful and -1 otherwise. Recall that the function requires a pointer to the head pointer in order to correctly delete the first node.

Write a program to test the removeNode() function. It should first allow the user to create a linked list of integers by appending values to the end of the list. Next, it should allow the user to delete nodes one by one based on their indices.

2. Re-write the function removeNode() using the LinkedList struct defined in the lecture materials.

```
int removeNode2(LinkedList *ll, int  index);
```

3. Write a function split() that copies the contents of a linked list into two other linked lists. The function prototype is given below:

```
split(ListNode *head,
      ListNode **ptrEvenList,
      ListNode **ptrOddList);
```

The function copies nodes with even indices (0, 2, 4, etc) to evenList and nodes with odd indices (1, 3, 5, etc) to oddList. The original linked list should not be modified.

Sample output:

```
Current list: 1 3 5 2 4 6 19 16 7
Even list: 1 5 4 19 7
Odd list: 3 2 6 16
```

4. Write a function duplicateReverse() that creates a duplicate of a linked list with the nodes stored in reverse. The function prototype is given below:

```
int duplicateReverse(ListNode *head,
                     ListNode **ptrNewHead);
```

The function should return 0 if the operation was successful and -1 otherwise. newHeadPtr should point to the first node of the reversed duplicate list.

Sample output:

```
Current list: 1 3 5 2 4 6
Reversed list: 6 4 2 5 3 1
```