```c
#include <stdio.h>
#include <stdlib.h>

typedef struct _listnode{
    int item;
    struct _listnode *next;
} ListNode;


void printList(ListNode *cur);
ListNode * findNode(ListNode *cur, int index);
int insertNode(ListNode **ptrHead, int index, int item);
void deleteList(ListNode **ptrHead);
int duplicateReverse(ListNode *cur,ListNode **ptrNewHead);

int main()
{
    ListNode *head=NULL;
    ListNode *dupRevHead=NULL;
    int size =0;
    int item;

    printf("Enter a list of numbers, terminated by any non-digit character: \n");
    while(scanf("%d",&item))
        if(insertNode(&head,size, item)) size++;
    scanf("%*s");

    printf("\nBefore duplicateReverse() is called:\n");
    printList(head);

    duplicateReverse(head,&dupRevHead);

    printf("\nAfter duplicateReverse() was called:\n");
    printf("The original list:\n");
    printList(head);
    printf("The duplicated reverse list:\n");
    printList(dupRevHead);

    if(head!=NULL)
       deleteList(&head);
    if(dupRevHead)
       deleteList(&dupRevHead);

    return 0;
}

void printList(ListNode *cur){
    printf("Current List: ");
    while (cur != NULL){
        printf("%d ", cur->item);
        cur = cur->next;
    }
    printf("\n");
}

ListNode *findNode(ListNode* cur, int index)
{
   if (cur==NULL || index<0)
      return NULL;
   while(index>0){
      cur=cur->next;
      if (cur==NULL)
         return NULL;
      index--;
   }
   return cur;
```

```c
}

int insertNode(ListNode **ptrHead, int index, int item){
    ListNode  *pre, *newNode;
    // If empty list or inserting first node, update head pointer
    if (index == 0){
        newNode = malloc(sizeof(ListNode));
        newNode->item = item;
        newNode->next = *ptrHead;
        *ptrHead = newNode;
        return 1;
    }
    // Find the nodes before and at the target position
    // Create a new node and reconnect the links
    else if ((pre = findNode(*ptrHead, index-1)) != NULL){
        newNode = malloc(sizeof(ListNode));
        newNode->item = item;
        newNode->next = pre->next;
        pre->next = newNode;
        return 1;
    }
    return 0;
}

void deleteList(ListNode **ptrHead){
    ListNode *cur = *ptrHead;
    ListNode *temp;
    while (cur!= NULL) {
        temp=cur->next;
        free(cur);
        cur=temp;
    }
    *ptrHead=NULL;
}

int duplicateReverse(ListNode *head,ListNode **ptrNewHead)
{
/* Write your program code here*/
    ListNode *cur = head;

    if (cur == NULL)
        return -1;

    // Traverse the linked list:
    while(cur != NULL)
    {
        if (insertNode(ptrNewHead, 0, cur->item) == -1)
            return -1;
        cur = cur ->next;
    }
    return 0;
}
```