Worksheet 17 ANSWER: Linked List Introduction, List Stack

```
#include <assert.h>
#include <stdlib.h>
#include "linkedListStack.h"

/* Singly Linked List Link Structure */
struct Link {
	TYPE		val;
	struct Link *next;
};


/* Singly Linked List with firstLink only */
struct LinkedList {
	struct Link *firstLink;
};




/*
	initLinkedList
	param: l the linked List
	pre: l is not null
	post: the firstLink of the linked list is initialized to null, isEmpty returns true
*/
void initLinkedList(struct LinkedList *l)
{
  l->firstLink = NULL;
}


/* LinkedListCreate
 pre: none
 post: none
 return: newly allocated Linked List structure
 */

struct LinkedList *createLinkedList()
{
	struct LinkedList *newList = malloc(sizeof(struct LinkedList));
```

```
        initLinkedList(newList);
        return newList;
}

/*
        linkedListFree
        param: l the linked list
        pre: l is not null
        post: sizeLinkedList returns true
*/

/*----------------------------------------------------------------------*/
/* Note: Free gets rid of all links but keeps the firstLink of the list around, so
  the list itself still exists and is initialized.
*/

void _freeLinkedList(struct LinkedList *l) {
 while (!isEmptyLinkedList(l))
  {
   popLinkedList(l);
  }
}

void deleteLinkedList(struct LinkedList *l)
{
        _freeLinkedList(l);
        free(l);
}

/*
        isEmptyLinkedList
        param: l the linked list
        pre: l is not null
        post: none
*/
int isEmptyLinkedList(struct LinkedList *l)
{
  return (l->firstLink == NULL);
}

/* Stack Interface */

/*
        pushLinkedList
        param: l the linked list
        param: val the value to be pushed
        pre: l is not null
        post: l is not empty, l size has increased by one
*/

/*---------------------------------------------------------------------------*/
void pushLinkedList(struct LinkedList *l, TYPE val) {
  struct Link *link = (struct Link *)malloc(sizeof(struct Link));
  assert(link != NULL);

  link->next = l->firstLink;
  link->val = val;
  l->firstLink = link;
}

/*
        topLinkedList
        params: l the linked list
        pre:  l is not null
        pre: l is not empty
        post: none
*/

/*---------------------------------------------------------------------------*/
TYPE topLinkedList(struct LinkedList *l) {
  assert(!isEmptyLinkedList(l));
  return l->firstLink->val;
}

/*
        popLinkedList
        param: l the linked list
        pre: l is not null
        pre: l is not empty
        post: l size has decremented by one
*/
/*---------------------------------------------------------------------------*/
void popLinkedList(struct LinkedList *l) {
  struct Link *link = l->firstLink;
  assert(!isEmptyLinkedList(l));

  l->firstLink = link->next;
  free(link);
}
```