Lab 9 - section F

We'll assume the following node structure:

```
typedef struct nodestruct {
    int item;
    struct nodestruct *next;
} Node;

typedef struct {
    int size;
    Node *head, *tail;
} List;
```

In this lab, you need to implement six functions based on the given structure definition above with no dummy nodes in the linked list:

*List *initIntegerList(); // Return empty list, O(1) complexity*
int insertAtTail(int, List); // Insert a new key at tail, O(1) complexity*
int removeTail(List); // remove the tail node from the list and return its key, O(n) complexity*
void emptyList(List); // This function receives a linked list, and removes all nodes in this list such that the list becomes empty, O(n) complexity*
*void freeList(List**); // This function receives the reference of the current linked list, frees all previously allocated memories in the linked list and set the list pointer to NULL, O(n) complexity*
void printList(List); // print out the current linked list, O(n) complexity*

Main program steps:

1. Create an empty linked list and print out the current linked list.
2. Insert 5 random integers with values between 1 and 10 into the linked list using function "insertAtTail". After each insertion, print out the current linked list.
3. Remove the nodes at tail twice by using function "removeTail". After each deletion, print out the current linked list.

4. Empty the linked list by using “emptyList” function.

5. Insert one random integer with value between 1 and 10 into the linked list using function “insertAtTail”. After this insertion, print out the current linked list.

6. Free your linked list by using “freeList” function. Make sure to not leave any memory leak.

Example output:

```
List is empty
Tail insertion: List is: 9
Tail insertion: List is: 9 8
Tail insertion: List is: 9 8 4
Tail insertion: List is: 9 8 4 6
Tail insertion: List is: 9 8 4 6 7
Tail deletion: List is: 9 8 4 6
Tail deletion: List is: 9 8 4
List has been emptied
Tail insertion: List is: 3
List has been freed
```

Grading Criteria:
Main program: 5 points
initIntegerList function: 5 points
insertAtTail function: 5 points
removeTail function: 10 points
freeList function: 5 points
emptyList function: 5 points
printList function: 5 points
All or nothing rule when grading: starting from lab 6, no partial credits will be given for incomplete/incorrect functions.

General note:

1. Command to compile your code in cmd window: `gcc labx.c -Wall -Werror`

2. If your code does not compile with “-Wall -Werror” flag, you will receive an automatic 0 for this assignment.
3. Changing the given function prototype or struct definition will lead to an automatic zero grade.
4. Using any global variables will lead to an automatic zero grade.
5. The implementation of the function should include comments describing what it is intended to do and how this function should be called. Example can be found in CS 2050 lab policy.
6. If your submission does not include a source file, you will receive an automatic zero grade.