Lab 8 - section F

We'll assume the following node structure:

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

In this lab, you need to implement six functions based on the given structure definition above with dummy nodes on both head and tail:

```
Node *initIntegerList();
int insertAtTail(Node*,int);
void printList(Node*);
void freeList(Node*);
int insertDescend(Node*,int);
int computeOdd(Node*);
```

The first four functions follow the same instruction as the prelab.

"insertDescend" function inserts a new node into the linked list based on the integer values of the linked list in descending order (see example output). This function returns 0 if the insertion was successful and -1 if the insertion fails.

"computeOdd" function counts the number of odd integers in your linked list and returns the count as an integer.

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 "insertDescend". After each insertion, print out the current linked list and the current number of odd integers in the linked list.

3. Insert another 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 and the current number of odd integers in the linked list.

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

Example output:
Linked list is empty.
0 odd numbers are in this linked list.
Descending insertion: Linked list is: 9
1 odd numbers are in this linked list.
Descending insertion: Linked list is: 9 2
1 odd numbers are in this linked list.
Descending insertion: Linked list is: 9 3 2
2 odd numbers are in this linked list.
Descending insertion: Linked list is: 9 3 3 2
3 odd numbers are in this linked list.
Descending insertion: Linked list is: 10 9 3 3 2
3 odd numbers are in this linked list.
Tail insertion: Linked list is: 10 9 3 3 2 5
4 odd numbers are in this linked list.
Tail insertion: Linked list is: 10 9 3 3 2 5 5
5 odd numbers are in this linked list.
Tail insertion: Linked list is: 10 9 3 3 2 5 5 4
5 odd numbers are in this linked list.
Tail insertion: Linked list is: 10 9 3 3 2 5 5 4 7
6 odd numbers are in this linked list.
Tail insertion: Linked list is: 10 9 3 3 2 5 5 4 7 7
7 odd numbers are in this linked list.
Linked list has been freed

Grading Criteria:
Main program: 5 points
initIntegerList function: 5 points
insertAtTail function: 5 points
insertDescend function: 10 points
printList function: 5 points
freeList function: 5 points
computeOdd 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.