CS201 Spring 2019
Assignment Two: singly-linked list
Part I (5 points): due Monday, Feb. 4th at 11:59pm
Part II (5 points): due Friday, Feb. 8th at 11:59 pm

Use this structure:

```
typedef struct StudentListNodeStruct {
  int id;
  char name[32];
  struct StudentListNodeStruct *next;
} StudentListNode;
```

and write functions to maintain a singly-linked list.

Here are four functions to write:

`int insertStudent(StudentListNode **list, int id, char *name)`

This will insert a new record in the list, sorted (ascending) by student id. Check to see whether there is already a record in the list having that id. If there is, don't insert a new record and return 1. Otherwise, create a new list node, put the list node in the correct place in the list, and return 0.

`int findStudent(StudentListNode *list, int id, char *name)`

Find the record having the specified id. If there is one, then copy the name from that record to the name parameter and return 0. Otherwise, just return 1.

`int deleteStudent(StudentListNode **list, int id)`

Delete the record from the list having an id that matches the specified id. If there isn't one, then just return 1. Otherwise, delete the record from the list and return 0.

`int printList(StudentListNode *list)`

Print the records from this list. Print them one to a line, in in this form:

```
|id name|
|id name|
|id name|
```

If the list is empty, then print this:

```
(empty list)
```

Part I

For insertion, there are three cases:
1. insert into empty list
2. new node goes in the middle of the list
3. new node goes at the front of the list

For deletion, there are three cases:
1. delete first node in list
2. delete node from middle of list
3. delete last node in list

For each of the cases for insertion and deletion, describe, using pseudocode, what the function needs to do. Make sure you describe what has to happen to the "next" pointers for list nodes for each of the three cases.

Part II

Do the actual implementations in C of the four functions above. Put your structure definition and function prototypes in slist.netid.h, and put your code in slist.netid.c

Run the tests in slists-tests.c (from gitlab).

Here's the gitlab repo for the class:

https://gitlab.uvm.edu/Jason.Hibbeler/ForStudents/tree/master/CS201/

Compile your code either this way:
$ gcc -c slist.netid.c slist-tests.c
$ gcc list.netid.o slist-tests.o

or this way:
$ gcc slist.netid.c slist-tests.c

Here's the output you should get:

```
inserted Trey Burke
inserted Tim Hardaway Jr.
inserted Emmanuel Mudiay
inserted Lance Thomas
inserted Alonzo Trier
failed to insert Sterling Brown
|1 Emmanuel Mudiay|
|3 Tim Hardaway Jr.|
|14 Alonzo Trier|
|23 Trey Burke|
|42 Lance Thomas|
found student 1: Emmanuel Mudiay
did not find student with id = 2
found student 42: Lance Thomas
```

```
did not find student with id = 100
found student 23: Trey Burke
deleted 14
deleted 1
found student 3: Tim Hardaway Jr.
failed to delete 6
deleted 3
deleted 42
failed to delete 1
deleted 23
(list is empty)
did not find student with id = 1
```