Assignment #2: Singly-Linked List

CS201 Fall 2019
10 points, due Monday, Sept 9th, 11:59 pm

1 Singly-Linked List

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. When you insert, first 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)

2 What to Do

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`; submit these two files to Blackboard.

NOTE: you should not have a `main()` — only the four functions I describe above.

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 John
inserted Elizabeth
inserted Franklin
inserted Petunia
inserted Archimedes
failed to insert Phoebe
1 |Franklin|
3 |Elizabeth|
14 |Archimedes|
23 |John|
42 |Petunia|
found student 1:  Franklin
did not find student with id = 2
found student 42:  Petunia
did not find student with id = 100
found student 23:  John
deleted 14
deleted 1
found student 3:  Elizabeth
failed to delete 6
deleted 3
deleted 42
failed to delete 1
deleted 23
(list is empty)
did not find student with id = 1
```

3 Extra Credit: Doubly-Linked List

Use this data structure:

```
typedef struct StudentDListNodeStruct {
  int id;
  char name[32];
  struct StudentDListNodeStruct *next;
  struct StudentDListNodeStruct *prev;
```

```
} StudentDListNode;
```

and implement these four functions:

```
int insertStudent(StudentDListNode **list, int id, char *name);
int findStudent(StudentDListNode *list, int id, char *name);
int deleteStudent(StudentDListNode **list, int id);
int printList(StudentDListNode *list);
```

This will be for a maximum of 10 extra-credit points. When we check your code, we will really bang on it, so pay attention to your pointers!

If you choose to do this, submit two files: `dlist.netid.c` and `dlist.netid.h`.