CS 2401 Spring 2023

Lab 5: Linked Lists

Due: Friday 2/24 at 11:59 PM

Begin by making a new directory for this lab, and then open a file called `list.h`. Into this file type the following code (please do type it on your own, it will help you learn it better and you can accidentally insert hidden characters if you copy and paste):

```
#include <iostream>

#include <string>


struct Node{

    std::string data;

    Node *next;

};


class Lilist{

    public:

        Lilist(){head = NULL;}

        void add(std::string item);

        void show();

    private:

        Node *head;

};
```

```
void Lilist::add(std::string item){
    Node * tmp;
    if(head == NULL){
        head = new Node;
        head -> data = item;
        head -> next = NULL;
    }
    else{
        for(tmp = head; tmp -> next != NULL; tmp = tmp -> next)
            ;  // this loop simply advances the pointer to last node in
               //the list
        tmp -> next = new Node;
        tmp = tmp -> next;
        tmp -> data = item;
        tmp -> next = NULL;
    }
}

void Lilist::show(){
    for(Node *tmp = head; tmp != NULL; tmp = tmp -> next)
        std::cout << tmp -> data << "  ";
}
```

Now write a main that looks like this in a file called main.cc

```
#include <iostream>

#include <string>

#include "list.h"

using namespace std;


int main(){

    Lilist L1, L2;

    string target;

    L1.add("Elizabeth");

    L1.add("Zachary");

    L1.add("Clay");

    L1.add("Lainie");

    L1.add("Izaak");

    L1.add("Andrew");

    cout << "Now showing list One:\n";

    L1.show();

    // END OF PART ONE

    /* cout << "Enter a name to search:";

    cin >> target;

    if(L1.search(target) != NULL)

           cout << "That name is stored at address: "

                << L1.search(target) << endl;

    else{

          cout << ”That name is not in the list.\n”;

    }

    L1.move_front_to_back();

    L1.move_front_to_back();

    L1.show();

    */

    return 0;

}
```

After you have written and run the program down to the place where it says End of Part One, go back into your class, and write a `search` function that takes a string as its argument. It will return the address of the node holding the string, or `NULL` if the pointer runs off the end of the list. *Hint*, part of this code will include:

```
  if(cursor -> data == target) return cursor;
```

Then write a `move_front_to_back` function. The key here is to:

1. Use an extra pointer to hold the first node in the list.
2. Move the head to the second node of the list.
3. With another pointer, find the last node in the list.
4. Hook the node that used to be at the front to the back. (Look at the code you've written for adding nodes).

Uncomment the rest of the main.

When you are done, run the program with a script file:

```
script myresults

./a.out

ctrl-d
```

If you do not have the script command available, take and submit screenshots of your terminal showing the complete output.

This file will show the list printed both in its original order and in the order with the first two names moved to the back of the list. Submit this script file along with your two source code files to Blackboard.