Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

Students:

Section 3.20 is a part of 1 assignment: **Programming Assignment 1**

This assignment's due date has passed. Activity will still be recorded, but will not count towards this assignment (unless the due date is changed). See this article for more info.

Requirements: zyLab

Entire class was due: Due: 08/04/2021, 11:59 PM PDT

3.20 Programming Assignment 1 - Singly-linked List (Summer B 21)

Due Tuesday Aug 3, 2020 at 11:59 PM

Collaboration Policy Reminder

You may not use code from any source (another student, a book, online, etc.) within your solution to this PROGRAM. In fact, you may not even look at another student's solution or partial solution to this PROGRAM to use as a guide or otherwise. You also may not allow another student to look at any part of your solution to this assignment. You should get help on this assignment by coming to the instructors' office hours or by posting questions on Slack. Do not post your code on Slack.

Problem Definition

You will implement singly-linked list in a MyList class. Your class definition will be implemented in a MyList.cpp. Each function's expected behavior is described below. You will find the following files in the specs: Node.h, MyList.h, and main.cpp (for early testing). Node.h is read-only in both develop and submit mode of this assignment. The main.cpp and MyList.h

A reasonable plan of attack is to implement and test the more straight-forward functions (constructors, destructor, push_back etc.) prior to working on the more challenging parts (find, swap, reverse). Your goal should be to complete a portion of the program each day. You have one-week to complete this assignment. Start early!

Node.h

There is no need to make any changes to this file either for debugging or testing. It is provided as read-only in both develop and submit mode.

```
#ifndef __NODE_H_
#define __NODE_H_

class Node {
  friend class MyList;
private:
  Node* next;
  char value;
};
#endif /* NODE_H_ */
```

MyList.h

You can comment out sections of MyList.cpp in developer mode. This will allow piece-wise compilation and testing of your solution as it is developed.

```
#include "Node.h"
#include <iostream>

using namespace std;

#ifndef MYLIST_H_
#define MYLIST_H_

class MyList{
 private:
  Node* head;
```

```
  void pop_front();
  void pop_back();
  void swap(int i, int j);
  void insert_at_pos(int i, char value);
  void reverse();

  /* Accessors */
  int  size()const;
  void print()const;
  int  find(char value)const;
  int  find(MyList& query_str)const;

  /* Operator overloaders */
  MyList& operator=(const MyList& str);
  char& operator[](const int i);
  MyList operator+(const MyList& str);

  /* Newly assigned functions */
  bool is_palindrome()const;
  void merge_list(MyList A, MyList B);
  bool remove_char(char c);
  char front()const;
};
#endif /* MYLIST_H_ */
```

Constructors and Destructor

MyList();

Default constructor

MyList(const MyList& str);

Copy constructor. Constructs a list from a passed in MyList object, e.g. MyList l1(l2);. Performs

Inserts value at the front of the list.

void push_back(char value);

Inserts value at the back of the list.

void pop_front();

Removes the first item from the list. No action on empty list.

void pop_back();

Removes the last item from the list. No action on empty list.

void swap(int i, int j);

Swaps the value of the nodes at positions i and j. Program should take no action if i or j is out-of-bounds.

void insert_at_pos(int i, char value);

Inserts a node with value at position i in the list. Program should abort via an assert statement if i is out-of-bounds.

void reverse();

Reverses the items in the list.

Accessors

int size() const;

Returns the number of nodes in the list.

void print() const;

Overloaded [ ] operator. Reads the character located at l1[i] in list l1. Program should abort via an assert statement if i is out-of-bounds.

char& operator[ ](const int i);

Overloaded [ ] operator. Returns writable reference to memory location for list l1 at l1[i]. Program should abort via an assert statement if i is out-of-bounds.

NEWLY ASSIGNED FUNCTIONS

bool is_palindrome()const;

Checks whether list object (implicit) contains a palindrome.
https://www.merriam-webster.com/dictionary/palindrome

void merge_list(MyList A, MyList B);

Merges two sorted lists A and B into implicit list object. HINT: A and B are passed-by-copy.

bool remove_char(char c);
Removes all instances of char c from implicit list object, e.g. remove all commas (',').

char front()const;
Returns the first node's value.

main.cpp

```
l1.push_front('0');
l1.push_front('z');
l1.print();

int loc_size = l1.size();
cout << "array style print: " ;
for(int i = 0; i < loc_size; i++)
  cout << l1[i] << " ";
cout << endl;

cout << "insert_at_pos a 0 on left and right of middle z: " ;
l1.insert_at_pos(2, '0');
l1.insert_at_pos(5, '0');
l1.print();

cout << "push back letters f-j: ";
```

```
MyList l2(l3);

// test pop_back
cout << "pop_back entire list, l1: ";
loc_size = l1.size();

for(int i = 0; i < loc_size; i++)
  l1.pop_back();
l1.print();

// test reverse
```

```
    cout << "list l4 + l1: ";
    l4 + l1;
    l4.print();

    cout << "Good Bye!" << endl;
    return 0;
```