CSE 310 Assignment #1

(Max. Points: 30)

Due on: Friday, Sep. 1st, 2017, 5:30pm Arizona time

General Instructions:

- This is an individual assignment, please do not collaborate. Make sure that you write every line of your own code. Using code written by someone else will be considered a violation of the academic integrity and will result in a report sent to the Dean's office.
- It must be submitted online through Blackboard submission link! We don't accept submissions through emails.
- Submission link will be closed automatically once the due date/time is past and **no late assignment will be accepted**.
- You will be allowed 3 times to submit the assignment, but we will only grade your last submission.

Objectives

- Review on C++ and some basic data structures such as linked lists.
- Review on dynamic memory allocation
- Review on basic Linux commands, know how to archive and compress files or create a make file.

Given File(s) in C++

Assignment1.cpp (No need to be modified)
LinkedList.h (More code need to be added)

Assignment Description

Write a C++ program that creates a linked list of *Country*. *Country* is a structure defined as below:

```
struct Country
{
    string name;
    int gold;
    int silver;
    int bronze;
    struct Country * next;
};
```

You need to create a pointer ("*head*") that points to the first element in the linked list (initially it is set to NULL). Also, you need to create the following functions in the ***LinkedList*** class (within LinkedList.h file):

`bool LinkedList::addCountry(string countryName, int gold, int silver, int bronze)`

- This method adds a new country data into the linked list by using the parameter values – country name, the number of gold, silver and bronze medals. It needs to create an object of the struct *Country* and add it inside the linked list at a correct location. Each country should be added in alphabetical order of their names. If there is no enough memory left to create the new object, the function should return *false*; otherwise the function returns *true* (*i.e.* the new country is successfully added).

`bool LinkedList::removeCountry(string countryName)`

- This method removes the country with the parameter country name value and should return *true* if it can find and remove it successfully; if countryName doesn't exist in the linked list, it returns false.

`bool LinkedList::changeMedalCount(string countryName, string medal, int count)`

- This method changes the country with the parameter country name value with the parameter medal count and should return *true* if it find and update the country info. successfully; otherwise it return *false* if countryName does not exist in the liked list. Note: the string parameter "*medal*" will be one of "*gold*", "*silver*", or "*bronze*".

`void LinkedList::printList( )`

- It prints all entries in the linked list in the following format (print "`The list is empty\n`" if the linked list is empty):

```
Country name: Hungary, Gold: 2, Silver: 3, Bronze: 0
Country name: Italy, Gold: 2, Silver: 5, Bronze: 4
Country name: South Korea, Gold: 3, Silver: 4, Br onze: 3
Country name: United States, Gold: 3, Silver: 5, Bronze: 6
```

Note that above countries were printed in alphabetical order.

`LinkedList::~LinkedList( )`

- This is a destructor of the *LinkedList* class. It should remove all elements in the linked list and perform garbage collection (free their memory). At the end, it should print: *"The number of deleted countries is: ??\n"*. where ?? is the number of deleted countries in the linked list.

`int main()`

- The main function is inside Assignment1.cpp, it should start by displaying the following menu:

Choice\t\t Action\n
------\t\t ------\n
A\t\t Add Country\n
D\t\t Display Countries\n
M\t\t Change Medal Count\n
Q\t\t Quit\n

R\t\t Remove Country\n
? \t\t Display Help\n\n
Next, the following prompt should be displayed:

What action would you like to perform?\n

Read in the user input and execute the appropriate command. After the execution of each command, re-display the prompt. Commands should be accepted in both lowercase and uppercase.

Sample Run

Click here to see a sample run of the program.

Other Requirements

- You need to implement this program by using C++ and it has to read from the standard input (from a keyboard).
- Your program needs to compile and execute in *general.asu.edu*.
- Your code should be well documented and commented.
- You are not allowed to use any predefined data structures (such as ones in the library, etc.) except arrays and strings, you need to build your own data structures and operations associated with them (such as add and remove).
- Copying or using any code from other people's programs is considered to be cheating and will lead to a report to the Dean and you will be penalized. (Please review the syllabus)

- At the top of each source code file, add the following information:

```
// Name of Author(s):
// ASU ID:
// Homework Number: 1
// Description:
```

Input & Output Test Cases

Click here to see one input and output test case, use it to check your program. We will test your program by using other test cases as well.

input1.txt SolutionOutput1.txt

After successfully compile your codes, test it by using above input test case, and save your output as output1.txt, such as follows. Then compare your output1.txt with above SolutionOutput1.txt, they should be identical.

```
$ ./a.out < input1.txt >output1.txt
```

(*replace a.out with whichever the executable file is)

Submission

1. On *general.asu.edu*, create a directory called **cse310_hw1**, put *input1.txt*, *Assignment1.cpp*, *LinkedList.h* inside it.

2. Use ***vi*** to create the Make file called **Makefile** in **cse310_hw1** directory for execution. Specify the executable filename in the Makefile as **prog** (if you don't know how to do this, check the lecture notes and sample Make file posted on Blackboard, under "Lecture Notes" => "GNU/Linus Commands").

3. Type **make clean** in your source code directory to clean your object and executable files after you test it with the input test case. Then create a **bzipped tarball** named ***cse310_hw1_lastname.tar.bz2***, where lastname is your last name

4. Upload and submit your bzipped tarball file ***cse310_hw1_lastname.tar.bz2*** onto Blackboard before the deadline.

Grading Rubric

____/ 3 Documentation (header including a description and comments for each function, class and file)
____/ 1 Indentation and Spacing (easy to read)
____/ 2 the *addCountry* function is defined correctly (and it produces the correct output)
____/ 2 the *removeCountry* function is defined correctly (and it produces the correct output)
____/ 2 the *changeMedalCount* function is defined correctly (and it produces the correct output)
____/ 1 the *printList* function is function correctly (and it produces the correct output)
____/ 1 the destructor *~LinkedList* function is defined correctly (and it produces the correct output)
____/ 18 Correct output for the test cases

Total points: 30