CSc 300
Assignment #1
Gamradt
Due: 09-25-23 (Late: 10-02-23)

Create a user-defined Abstract Data Type (ADT) named **List**

- Use an appropriate set of C++ header/implementation files as discussed in class
- **List** is implemented as a **Single-Linked-List**
- **List** consists of 0 or more **Element** values
  - **Element** is an exportable standard primitive **float** data type
    - **Element** is managed using **dynamically allocated nodes – Node**
    - See **C++ Pointers** under D2L Lecture Notes
  - **Node** consists of two fields:
    - **element**
    - **next**

The **List** ADT must define and implement the following data types and operations

- Do not add to or modify the public interface (exportable components – public components)
- Do not add to or modify any attributes or data types (storage components)

Exportable Operations: (declared .h file and defined .cpp file)

List	default constructor function – creates an initialized empty list
List	copy constructor – creates a duplicate copy of an existing list reuse the add method when populating the new list
~List	destructor function – removes all elements from the list list instance state before going out of scope – initialized empty list
add	inserts a new element to the list in descending order allocation fails provide user feedback
remove	removes an existing element from the list not found provide user feedback
view	displays the contents of the list from the front of the list to the end of the list view function uses a non-destructive **iterative** implementation

User-Defined Data Types:
Element
Node
NodePtr

List Required Output Format: (view)

```
FRONT -> END                                // Output for an empty List instance
FRONT -> 10.00 -> 0.00 -> -10.55 -> END     // Output for a populated List instance
```

Required header file (.h).   **// only partially specified**

```
//  General description of the ADT and supported operations – exportable operations only
//  Do not include any implementation details

#ifndef _LIST_H                                         // Guard – start
#define _LIST_H

typedef float Element;                                  // typedef  <existing type>  <new type>
                                                        // basic form of generic programming
class List {
        public:                                         // exportable
//  General description of each of the ADT operations/methods/functions – exportable operations only
                List();
                List( const List & );                   // reuse add
                ~List();
                void add( const Element );
                void remove( const Element );
                void view() const;
        private:                                        // non-exportable
//  No private member documentation – implementation details are hidden/abstracted away
                struct Node;
                typedef Node * NodePtr;
                struct Node {
                        Element element;
                        NodePtr next;
                };
                NodePtr front;
};

#endif                                                  // Guard – end
```

List ADT include sequence: // Never include .cpp files

$$\begin{tikzcd}\text{main.cpp} \arrow[r] & \text{List.h} & \text{List.cpp} \arrow[l]\end{tikzcd}$$

List ADT incremental building sequence: // Using make

```
1.  Place all files in the project folder               // I would use Gamradt1
2.  make                                                // Process Makefile – generate executable
3.  ./output                                            // Run project
```

Make sure that you completely document the header/implementation files.

- The header (.h) file tells the user exactly how to use your ADT
  - General descriptions only – do not include implementation details
- The implementation file (.cpp) tells the implementer/programmer exactly how the ADT works
  - Detailed descriptions – include implementation details
- See **Documentation Requirements** – D2L Handouts Folder

I will write a test program that will include your **List** ADT so all header/implementation files tested must use common names. You **MUST** use:

- the **EXACT** same names for each data type and function in the header/implementation files
- the **EXACT** same function argument sequence in the header/implementation files

Use **PITA** everywhere possible

- Prefer Initialization to Assignment

Project Folder:	Lastname1	// I would use Gamradt1
• List.h	**List** class header file	
• List.cpp	**List** class implementation file	
• main.cpp	driver program file	// I will use my own
• Makefile	appropriate set of incremental build rules	// “1” module

Push your assignment solution to your GitHub account, then send me a access to the assignment repository

- E.g., CSc300 // CSc300
  - Remember that a 20% reduction is applied for not using GitHub
  - See **Assignment Requirements** – D2L Handouts Folder

List the class number, your lastname, and assignment number as the e-mail message subject:
SUBJECT: csc300 – Lastname – a1 // I would use “… Gamradt …”

Makefile **// Do Not Include Comments**
// “1” TAB required for indentation

```
output:  main.o  List.o                              // generate object files (.o)
	g++  -std=c++11  -o  output  main.o  List.o  // compile/build command

main.o:  main.cpp                                    // Rule for generating main.o file
	g++  -std=c++11  -c  main.cpp

List.o:  List.h  List.cpp                            // Rule for generating List.o file
	g++  -std=c++11  -c  List.cpp

clean:                                               // Rule for cleaning project
	rm  output  main.o  List.o                   // Remove executable and object files
```