PIC 10B Section 1 - Homework # 7 (due Wednesday, May 22, by 6 pm)

You should upload each .cpp and .h file separately and submit them to CCLE before the due date/time! Your work will otherwise not be considered for grading. Do not submit a zipped up folder or any other type of file besides .h and .cpp.
Be sure you upload files with the precise name requested of you and that it matches the names you used in your editor environment otherwise there may be linker and other errors when your homeworks are compiled on a different machine. Also be sure your code compiles and runs on Visual Studio 2017.

Organize your code well and at the end of this work, submit **.h** and **.cpp** files that, when compiled with a user's main routine and linked, will provide them with the functionality of a **Linked List** data structure storing **int**s.

At the end of this work you should submit a header file **List.h** that includes the declarations of all classes and functions and **List.cpp** that defines those classes and functions.

Do not submit a main routine for this homework!

LINKED LIST

In class and in the notes, many of the implementation details were provided for a **LinkedList** data structure. Your job in this homework is merely to make it more complete.

You should write your **LinkedList** class to store **int**s and some helper functions; everything you write should be in a namespace **pic10b**.
Your **LinkedList** class must have the following implemented:

- a **default constructor** to start empty;
- a **constructor** that takes an **std::initializer_list<int>** and begins by storing those items;
- a **copy constructor**;
- a **move constructor**;
- a **destructor**;

- **assignment operators** (both copy and move);
- **push_back** to append an **int** at the end;
- **pop_back** to pop off the end element, throwing an exception if the list is empty;
- **push_front** to prepend an **int** at the beginning;
- **pop_front** to pop off the first element, throwing an exception if the list is empty;
- **begin** member function overloaded on const, returning iterators to the beginning;
- **end** member function overloaded on const returning iterators to the past-the-end;
- **insert** accepting an **interator** and value to insert, with the value being inserted before the point where the iterator points;
- **erase** accepting an **iterator** and removing the pointed-to element;
- **size** returning the number of elements in the list;
- **sort** to sort the elements in ascending order from beginning to end;

plus **operator**<< should be overloaded to display all the elements of the list with spaces between them.

Your **iterator** and **const_iterator** classes must have the following implemented:

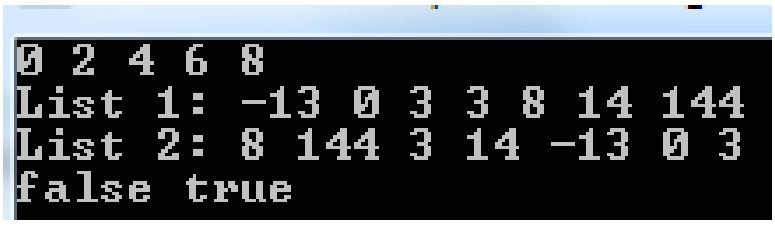
- **prefix and postfix ++ and --**;
- **operator==**;
- **operator!=**;
- **dereferencing** operator;

plus there should be a conversion from an **iterator** to **const_iterator**!

In addition to these classes, you should write a **seqSearch** function that accepts two **const_iterators** and a value; it should search over the range provided by the iterators from the first up to but not including the second iterator. If the sought value is found, it should return **true**; otherwise **false**.

In this homework **the only Standard Library header files that you are allowed to use are iostream and stdexcept**. Do not include any other header files.

See sample output (note that not all functions are demoed in this code so be sure to read the instructions carefully and include all the necessary member functions)




```
#include "List.h"

#include<iostream>

int main() {
  // uses initialzier list constructor
  pic10b::LinkedList list0{ 2,4,6,8 };
  list0.push_front(0);
  list0.push_front(-11);
  list0.pop_front();

  for (int i : list0) { // for all the ints, print them
    std::cout << i << " ";
  }
  std::cout << '\n';

  pic10b::LinkedList list1; // default constructor
  list1.push_back(8);
  list1.push_back(3);
  list1.push_back(14);
  list1.push_back(-13);
  list1.push_back(0);
  list1.push_back(3); // a second 3!!!

  auto it = list1.begin();
  ++it;

  list1.insert(it, 144);
  list1.insert(list1.end(), 44);
  list1.pop_back();

  const pic10b::LinkedList list2 = list1;

  list1.sort();

  std::cout << "List 1: " << list1 << '\n';
  std::cout << "List 2: " << list2 << '\n';
```

```
  auto beg1 = list1.begin();
  auto end1 = list1.end();
  end1--; // so it last element will not be searched through in the searche to come...

  auto beg2 = list2.begin();
  auto end2 = list2.end();

  bool found1 = pic10b::seqSearch(beg1, end1, 144);
  bool found2 = pic10b::seqSearch(beg2, end2, 144);

  // the boolalpha turns the 0/1 for false/true into "false" and "true"
  std::cout << std::boolalpha << found1 << " " << found2 << '\n';

  std::cin.get();

  return 0;
}
```

Some initial guidance to get you on your way...

1. Be sure to draw diagrams!
2. More than half of this work is provided to you in the lecture notes, so part of your work could just be transferring that over.
3. You may want to have an extra member variable to track the number of elements in your list (for the **size** function, especially).
4. There is no random access for the list and that means if you want to **sort** it, you may have to settle for one of those algorithms that's just $O(n^2)$... Just draw out your process out carefully and allow the iterators to manage the values you may want to swap.