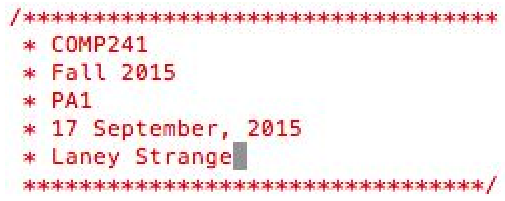
COMP241
Fall 2015
Programming Assignment 1
Assigned: 10 September, 2015
Deadline: 17 September, 2015
Early Deadline: 15 September, 2015: 5 points extra credit

Submission guidelines:

- This is your first programming assignment. Make sure you start early -- it might be harder than you think!
- Please post your solution on Moodle. It is due ***at the beginning of class.***
- For an extra 5 points, submit your solution on 15 September, at the beginning of class.
- Use MS Visual Studio for the assignment. Submit one .cpp file for each problem below. If requested, include accompanying text files with sample inputs and outputs separately.
- Include a comment with the course number, term, assignment, your name, and the date, like this:



```
/**********************************
 * COMP241
 * Fall 2015
 * PA1
 * 17 September, 2015
 * Laney Strange
 **********************************/
```

- In addition to writing excellent code, you will be graded on the user experience you create. Any time there are user prompts and input/output, make sure your code is both informative and polite.
- Although I encourage collaboration if you want to work with a classmate, you must turn in your own work. Duplicate code and plagiarism will be considered a violation of the honor code and handled accordingly.

Problem 1
20 points.

This problem will require you to implement your own version of an unsorted list. You will use a 1-dimensional array to represent the list. In fact, you'll just be working ***just*** with arrays for this one, not full-on classes. Don't overthink it!

Write a C++ program that determines the last element in a linked list of positive integers, implemented as a one-dimensional array.

You will implement a conceptual linked list with an array, represented as follows:

- A node of a linked list has two components: a value and a pointer.
- Any pair of elements in the array, given by `A[i]` and `A[i+1]`, collectively forms a node.
- For a given node, `A[i]` contains the value of the node, and `A[i+1]` points to the next element in the linked list.

- The `A[i+1]` pointer gives the the array index (not the memory address) of the next node. The element that it points to will always be a "value" element and not a "pointer" element.
- The array always begins with value/pointer pair `A[0]` and `A[1]`.
- A null pointer (signifying the end of the list) is represented by `-1`.

For example, suppose you have linked-list 18->12->5. One way to represent it with this scheme is:

```
A[0] = 18
A[1] = 4
A[2] = 5
A[3] = -1
A[4] = 12
A[5] = 2
```

Your program should accomplish three tasks:

1. Prompt the user for 8 nodes (16 total entries) matching the above linked-list scheme. The linked list you're creating should comprise positive integers; if the user enters a negative number apart from the -1 that declares the end of the list, nicely throw an error and skip that number.
2. Check that the input represents a well-formatted list. For example, if the user enters (18, 4, 5, 4), that's a problem because you get to the node at position 4 twice.
3. Call a function to repeatedly insert each node into an array representing the linked list.
4. Call a function to iterate through the list and print out the values as you go. For example, with the above linked list, you would print: *18,12,5*.
5. When the end of the linked-list is reached, print "*end of list at position X*" where X is the array index where the -1 was found. In the first linked-list above, this would read "end of list at position 3".

Include a text file along with your .cpp file submission that includes the following information:

- A list of arrays you've tried and tested, along with the results of running your program. To make sure your program works correctly, you must test with at least 2 different "good" lists and one"bad" (poorly formatted) one.

Problem 2
30 points

On Moodle, you will find links to `unsorted.h` and `unsorted.cpp`, which together define a class called `UnsortedList`. (They are also on gdrive: http://bit.ly/1Uzh6DF and http://bit.ly/1UEIFGq). It's similar to the example in the book, Chapter 3.2, but the data type is not abstract; rather, it's a class for an unsorted list of positive integers.

(Need a refresher on getting MS Visual Studio up and running? Check out Lab 0 from my COMP142 class.)

Write the code for a simple driver function, in a separate file called `driver.cpp`. Your driver should do the following:

- Create an UnsortedList object
- Attempt to add 4 integers to the list (note that the first 3 should succeed and the fourth should fail because MAX_ITEMS is set to 3). These 11 integers should be hard-coded or created via a loop in your code; do not prompt the user for them.
- Print the list. (There is no PrintList function, however; how can you use what you have without modifying the source code?)
- Delete two elements from the list. Ask the user which integers to delete.
- Print the list again.

For this problem, you don't need to modify unsorted.cpp or unsorted.h. If you *do* modify them, that's perfectly reasonable. Submit your driver.cpp to moodle, and if necessary, other modified files.

Problem 3
30 points
This problem also uses the `unsorted.cpp` and `unsorted.h` files.

Write a new function that takes two of your linked lists and merges them. This need not be a member function of the `UnsortedList` type, just a function that takes two `UnsortedList` objects.

Assume that both of your linked lists are in sorted order, and merge them into a single, sorted list. To test your merge code, make a driver that creates two sorted lists and calls the merge function.

For example, if you start with lists 5->10->18 and 2->3->17, then your merged list would be 2->3->5->10->17->18.

Don't make any assumptions about the relative size of your lists; they could be the same size as each other or totally different.

Again, you don't need to modify unsorted.cpp or unsorted.h. If you *do* modify them, that's perfectly reasonable. Submit your driver.cpp to moodle, and if necessary, other modified files.