These must be completed and shown to your lab TA either by the end of this lab session, or by the start of your next lab session.

Make a new sub-directory called “lab3” in your “cs221” directory, and download the lab3_files from the course web page into it.

Q1. Complete the missing code in “CDate.cc”. You do not need to change any other file for this question, but you need to understand what they do, and how they are used. The dot-h file (or “header”) contains the class declaration, and it's #include'd in both its associated dot-cc file, and any other files that need to use CDate objects (in this lab, just the main.cc file).

Read “main.cc”. Note the two different ways to construct a CDate object, and the extra statement that's used when the source is compiled under Windows.

The provided Makefile compiles and links these files by default, so to compile and run:

```
make
./CDate
```

If you complete the code correctly, you should see the following output:

```
1/1/2015
0/0/0
0/0/0
0/0/0
29/2/2000
0/0/0
31/12/2014
0/0/0
30/11/2010
0/0/0
29/2/2012
5/9/2014
```

Some helpful links: http://www.cplusplus.com/reference/string/string/compare/ and http://www.cplusplus.com/doc/tutorial/control/ (scroll down for the “switch” statement).

Q2. Complete the linked_list program. You will need to finish the following functions:

```
// This function deletes the last element in the linked list.
// Pre: The head of a linked list is provided.
// Post: The last element of that linked list has been removed.
void delete_last_element( Node*& head );

// If there is no Node with the given key, no action. Otherwise, insert a
// new Node (with key == newKey) after the node with the given key.
// Pre: The head of a linked list, a key to indicate where to insert, and
// the newKey to be inserted are provided.
// Post: If a node-with-key is found, the linked list now contains a new Node,
// with key EQ newKey, that is the successor of the node-with-key.
void insert_after( Node* head, int key, int newKey );

// This function creates a new linked list by merging two linked lists.
// Pre: Two linked lists (list1 and list2) are provided.
// Post: A new linked list is returned that contains the keys of both lists,
// starting with the first key of list1, then the first key of list 2, etc.
// When one list is exhausted, the remaining keys come from the other list.
// For example: [1, 2] and [3, 4, 5] would return [1, 3, 2, 4, 5]
Node* interleave( Node* list1, Node* list2 );
```

If you complete the code correctly, you should see the following output:

```
<A> List 1: [3, 2, 1]
<B> List 2: [6, 7, 8, 9, 10]
<C> List 1: [3, 2]
<D> List 1: [3]
<E> List 1: []
<F> List 1: []
<G> List 1: [11, 12]
<H> List 1: [11, 12]
<I> List 4: [11, 6, 12, 7, 8, 9, 10]
<J> List 4: [11, 12]
<K> List 4: []
```

Q3. (Recommended, but optional) If you used a recursive approach to implement the interleave method, create another implementation using an iterative approach. If you used an iterative approach, now use a recursive approach.