These must be completed and shown to your lab TA either by the end of this lab, or at the start of your next lab. You may work in groups of up to two people.

1. Complete the methods marked with TODO of the `CDate` class in `CDate.cc` (available under Lab 3 on the course web page). Look at CDate.h to give you an overview of the methods and instance variables. The only file you will need to change for this question is `CDate.cc`. Compile the code using `make` and run it using `./dates`. When you complete the code correctly, you should see the following output:

```
myDate0: 2015/1/1
myDate1: 0/0/0
myDate2: 0/0/0
myDate3: 0/0/0
myDate4: 2000/2/29
myDate5: 0/0/0
myDate6: 2014/12/31
myDate7: 0/0/0
myDate8: 2010/11/30
date9: 0/0/0
date10: 2012/2/29
date11: 2014/9/5
```

   Note that the test cases and `main()` function are located in `dates.cc`. Feel free to add more test cases to the bottom of the file, but don't make any changes to the ones that are already there.

   You can compare strings in C++ using the `==` operator. However, another way is to use the string member function `compare`. Examples of this, and other string functions, are found at:

```
http://www.cplusplus.com/reference/string/string/compare
```

   If you wish to learn more about the switch statement, you can do so here:

```
http://www.cplusplus.com/doc/tutorial/control/#switch
```

2. The second half of this lab deals with a recursive data structure called the Linked List. Complete the methods of the Linked List program (available under Lab 3 on the course web page). Compile using `make lists` and do an initial test run using `./lists`.

   When you complete the functions correctly, the tests will all pass. Drawing pictures of the possible configurations that must be handled in each method will help. You will need to complete the following functions:

```
void delete_last_element( Node*& head );

void remove( Node*& head, int oldKey);

void insert_after( Node* head, int oldKey, int newKey );

Node* interleave( Node* list1, Node* list2 );
```

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

```
<A> list1: [3, 2, 1]
<B> list2: [6, 7, 8, 9, 10]
<C> delete_last_element(list1): [3, 2]
<D> delete_last_element(list1): [3]
<E> delete_last_element(list1): []
<F> list1: [5, 10, 15]
<F> remove(list1,10): [5, 15]
<F> remove(list1,15): [5]
<F> remove(list1,5): []
<G> list1: [11]
<G> insert_after(list1,11,12): [11, 12]
<H> insert_after(list1,13,14): [11, 12]
<I> insert_after(list1,11,12): [11, 12, 12]
<J> delete_last_element(list1): [11, 12]
<K> interleave(list1,list2): [11, 6, 12, 7, 8, 9, 10]
<L> interleave(list2,list2): [6, 6, 7, 7, 8, 8, 9, 9, 10, 10]
<M> interleave(list1,NULL): [11, 12]
<N> interleave(NULL,list2): [6, 7, 8, 9, 10]
<O> interleave(NULL,NULL): []
```

3. Show your work to your TA either by the end of this lab or at the start of your next lab, or you will not receive credit for the lab!

4. **(Optional, not for marks)** 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.