1. Download List.java and ArrayList.java from ~dxu/handouts/labs/06. As we covered in class, the inner class ArrayIterator lets us create iterators over elements of the ArrayList. Study this class before you continue. The method iterator() of the ArrayList class creates and returns an Iterator.

2. Now, you will write another iterator class within the ArrayList class called MyListIterator. This iterator will be used to traverse the list in both forward and backward direction. next(), hasNext() methods are the same as in ArrayIterator and remove() is very similar. But it has additional methods as specified below:

   - boolean hasNext(): Returns true if this list iterator has more elements when traversing the list in the forward direction. (In other words, returns true if next() would return an element rather than throwing an exception.)
   - E next(): Returns the next element in the list and advances the cursor position. It throws NoSuchElementException if the iteration has no next element. This method may be called repeatedly to iterate through the list, or intermixed with calls to previous() to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)
   - boolean hasPrevious(): Returns true if this list iterator has more elements when traversing the list in the backward direction. (In other words, returns true if previous() would return an element rather than throwing an exception.)
   - E previous(): Returns the previous element in the list and moves the cursor position backwards. It throws NoSuchElementException if the iteration has no previous element. This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to next() to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)
   - void remove(): Removes from the list the last element that was returned by next() or previous(). This call can only be made once per call to next or previous. This method throws IllegalStateException if neither next nor previous have been called, or remove has already been called after the last call to next or previous. Refer to the remove() method in ArrayIterator which is very similar.
   - void set(E e): Replaces the last element returned by next() or previous() with the specified element. This call can be made only if remove() has not been called after the last call to next or previous. This method throws IllegalStateException if neither next nor previous have been called, or remove has been called after the last call to next or previous.

3. Add in your ArrayList class two methods myListIterator() and myListIterator(int i) that will make an instance of MyListIterator and position it at the beginning of the list, and position it right before the i-th index, respectively. These methods are similar to the iterator() method we wrote. Note that indices start with 0.
4. Write a driver class which creates and \texttt{ArrayList} of Integers, and creates a \texttt{MyListIterator} for it. Test ALL the methods in \texttt{MyListIterator}.

5. Now, in your driver, create two \texttt{ArrayList<Integer>} objects, called \texttt{L}, and \texttt{P}. Populate these lists with integers, \texttt{L} need not be sorted but \texttt{P} should contains integers sorted in ascending order (just initialize it that way, no need to sort). Call a method \texttt{removePositions(L, P)} that you write. \texttt{removePositions(L,P)} should remove the elements in \texttt{L} that are in positions specified by \texttt{P}. Positions start with 0. For instance, if \texttt{L} contains \{3, 10, 8, 5, 12, 67, 25, 22\} and \texttt{P} contains \{1, 3, 4, 6\}, the elements at index 1, 3, 4 and 6 in \texttt{L} should be removed, resulting in \texttt{L}=[3, 8, 67, 22]. Write the Java method \texttt{removePositions(ArrayList L, ArrayList P)}. You are REQUIRED to use your \texttt{MyListIterator} to implement \texttt{removePositions}. You may assume \texttt{P} only contains valid indices of \texttt{L}. 