The methods iterator() and positions() create and return an element iterator and an Iterable list of positions, respectively. In this homework, you will write another iterator class within the LinkedPositionalList class called ElementListIterator which implements Java's ListIterator interface. This iterator is an element iterator and will be used to traverse the list in both forward and backward direction and has additional methods as specified below:

- void add(E e): Inserts the specified element into the list at the current position of the iterator. In other words, the element is inserted immediately before the element that would be returned by next(), if any, and after the element that would be returned by previous(), if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

- 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.)

- int nextIndex(): Returns the index of the element that would be returned by a subsequent call to next(). (Returns list size if the list iterator is at the end of the list.)

- int previousIndex(): Returns the index of the element that would be returned by a subsequent call to previous(). (Returns -1 if the list iterator is at the beginning of the list.)

- 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. It can be made only if add(E) 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 or add have been called after the last call to next or previous.

- void set(E e): Replaces the last element returned by next() or previous() with the specified element. This call can be made only if neither remove() nor add(E) have been called after the last call to next or previous. This method throws IllegalStateException if neither next nor previous have been called, or remove or add has been called after the last call to next or previous.

Add in your LinkedPositionalList class two methods listIterator() and listIterator(int i) that will make an instance of ElementListIterator and position it at the beginning of the list, and position it right before the i-th index, respectively. Note that indices start with 0.

Write a driver to test all methods you wrote.