Project: DNA Splicing

Restriction enzymes represent a pattern in a DNA strand that can be split at a certain place with new DNA inserted. The original DNA strand is cut into two pieces with a new small strand inserted. In the real chemical process, the original strand may be split into several pieces at multiple binding sites. In some experiments, and in the simulation you'll run, the new DNA material will be spliced into the separated strand.

Consider an example where the original strand of DNA is "...CGAATTCG..." (see below). The letters C, G, A, and T represent Nucleotides, the molecules that, when joined together, make up the structural units of RNA and DNA. In this first example, the restriction enzyme (the pattern to search for) is EcoRI or "GAATTC". Your code will search for "GAATTC" (the restriction enzyme) in the original DNA strand and split that strand into two pieces. The new DNA "TGATA" is inserted somewhere in that recombinant enzyme. Consider this happening when one of the patterns is found:

1) Find the pattern representing the insertion point ("GAATTC" below)
2) index into the recombinant enzyme (if index == 1, go right once. This one will be cut after G)
3) insert the new DNA material ("TGATA" even further below).

![Diagram](image1.png)

The index of the split is one indicating that the first nucleotide/character of the restriction enzyme adheres to the left part of the split. The new DNA will be inserted after G in this case. The small DNA strand TGATA will now be spliced in like this.

![Diagram](image2.png)

When the spliced-in strand joins the split strand we see a new strand of DNA as shown below. The shaded areas indicate where the original strand was spliced by the new DNA material.

![Diagram](image3.png)

Your code will be a software simulation of this process: the restriction enzyme will be searched for and a small DNA strand will be spliced-in. Don't forget there is a number like 0, 1, or 3 to indicate how many nucleotides (letters G, A, T, or C) are to the left of the small spliced in strand.
**IDnaStrand** We have designed a new type to represent DNA strands with a method to simulate the chemical experiment. Given this interface IDnaStrand, complete the @Test method below to insert TGATA everywhere GAATTC occurs at index 1 GAATTC becomes GTGATAAATTCC exactly two times.

```java
public interface IDnaStrand {

    /**
     * Add all nucleotides in str to the end of this LinkedStrand (convert each letter
     * into a new node to be added at the end of the current Strand. This must run O(1).
     * Note: with a linked structure, maintain a reference to the last node.
     */
    public void append(String str);

    /**
     * Return the number of nucleotides (G, A, T, or C) in this strand.
     */
    public long size();

    /**
     * Return the nucleotides in the strand as a String such as "catgaatccat".
     */
    public String getStrand();

    /**
     * Perform a cut at each occurrence of patternToSearchFor and insert toBeSplicedIn the
     * correct number of characters into the patternToSearchFor. If the restrictionEnzyme
     * is not found, no cut or splice is made. The Strand remains the same.
     * Precondition: index < patternToSearchFor.length()
     */
    public void findPatternAndInsert(String patternToSearchFor, String toBeSplicedIn, int index);
}
```

```java
@Test
public void testOneSimulationWithTwoFindsAndOInsert() {
    IDnaStrand strand = new Strand(); // Assume Strand implements IDnaStrand

    To be arranged during Monday's lecture or section.

    Or write your own tests to gain an understanding of this process.
```
Implementation Specifics

We could use a String to store the letters representing the strands. However, studies have shown that computers cannot handle the size of this problem (you run out of memory) and the simulation runs slowly. Instead we will use a linked list with a reference to the first node to begin searches and a reference to the last node so we can append nucleotides (actually single characters) to the end of the singly-linked structure O(1).

You'll be developing/coding class LinkStrand that implements interface IDnaStrand (copy and paste above) Your class is to simulate cutting a strand of DNA when the given pattern is found splicing-in a new small strand, also supplied in the call to this method

```java
public void findPatternAndInsert(String patternToSearchFor,
                                 String toBeSplicedIn,
                                 int index);
```

You must use a singly-linked structure to support the operations--specifically the class LinkStrand should maintain references to a singly-linked structure used to represent a strand. You should keep and maintain a reference to the first Node of the singly-linked structure and to the last node of the linked structure. Store single chars in the node. No generics needed.

![Diagram of linked list structure](image)

Grading Criteria

___ / 90 points WebCat
___ / 10 Style and design

Ways to lose points. Please read

-80 You did not use a singly linked list as your data structure
-20 You do not maintains last reference to have append be O(1)
-20 The data in each node is something other than char
-90 You code did not compile and Webcat gives you 0 for unknown symbols
-90 Our tests exposed an infinite in your code WebCat times out and gives you 0
-90 You tests do not pass when turned in. WebCat gives you 0 for this sometimes