Programming Exercise: Finding a Gene and Web Links

Assignment 1: Finding a Gene

A **gene** has a start codon, followed by a substring that is a multiple of 3, followed by a stop codon. Write a program that 1) finds and prints a gene in a strand of DNA by using the algorithm below, and 2) also prints the stop codon that identifies the gene. If there is not a gene in the DNA string, then both should print the empty string. See below for more details.

Algorithm to identify a gene in a strand of DNA:

1. Identify the first start codon ATG in the string.
2. Identify the first occurrence of the stop codon TAG after the start codon. If the length of the substring between the start and stop codon is a multiple of three, the gene is the string from the beginning of the start codon to the end of the TAG stop codon.
3. If no gene is found yet, then identify the first occurrence of the stop codon TGA after the start codon. If the length of the substring between the start and stop codon is a multiple of three, the gene is the string from the start codon to the TGA stop codon.
4. If no gene is found yet, then identify the first occurrence of the stop codon TAA after the start codon. If the length of the substring between the start and stop codon is a multiple of three, the gene is the string from the start codon to the TAA stop codon.
5. If no gene found yet, then the gene returned is the empty string.
6. If a gene was found, then print the gene, and also print the stop codon that was used to find the gene.
7. Your program should work regardless of whether the DNA strand uses uppercase or lowercase letters.

To solve this problem, you should do the following:

- Make a copy of the `TagFinder` program that goes with this lesson. In particular start with the methods `findProtein` and testing methods. This program will be easier to debug if you use small examples.
- Modify the `findProtein` method to use the algorithm above to identify and return a gene for a protein.
- Use the testing method to test `findProtein`. You should create your own test strings and be sure to test examples that use all three stop codons.
- Write a method `stopCodon` that has one parameter that is a String representing a gene. This method returns the stop codon of the gene. It should return the empty string if the parameter is not a gene.
- In the testing method call `stopCodon` to print the stop codon found in the gene.

Here are some example strings you might want to test:

- "`AATGCTAGTTTAAATCTGA`"—This has all three stop codons. Using the algorithm above, the first start codon ATG is found. The stop codon TAG is found first but is not a multiple of three. The stop codon TGA is found next and is a multiple of three away, so "`ATGCTAGTTTAAATCTGA`" is the string returned.
- "`ataaactatgttttaaatgt`"—Using the algorithm above, the first occurrence of ATG is found. Then TAG is not found, TGA is not found, but TAA is found and is a multiple of three from ATG. The gene "`atgttttaa`" is returned. Note that TAA occurs twice in the string. Your program should look for the first one that occurs after the start codon in the string.
- "`acatgataacctaag`" returns ""—Note the first TAA is found and does not work. A second TAA is a multiple of three away, but our algorithm only looks for the first occurrence of TAA immediately past the start codon.

Assignment 2: Finding Web Links

Write a program that reads the lines from the file at this URL location, http://www.dukelearntoprogram.com/course2/data/manylinks.html, and prints each URL on the page that is a link to youtube.com.

Assume that a link to youtube.com has no spaces in it and would be in the format (where `[stuff]` represents characters that are not verbatim):

```
"http:[stuff]youtube.com[stuff]"
```

Here is a suggested algorithm:

1. Use URLResource and read the file word by word.
2. For each word, check to see if "youtube.com" is in it. If it is, find the double quote to the left and right of it to identify the URL. Note, the double quotation mark is a special character in Java. To look for a double quote, look for (\"), since the backslash (\) character indicates we want the literal quotation marks (") and not the Java character. The string you search for would be written `"\""` for one double quotation mark.
3. In addition to the String method `indexOf(x, num)`, you might want to consider using the String method `lastIndexOf(s, num)` that can be used with two parameters `s` and `num`. The parameter `s` is the string or character to look for, and `num` is the last position in the string to look for it. This method returns the last match from the start of the string up to the `num` position, so it is a good option for finding the opening quotation mark of a string searching backward from "youtube.com." More information on String methods can be found in the Java documentation for Strings: http://docs.oracle.com/javase/7/docs/api/java/lang/String.html.

Caution: The word Youtube could appear in different cases such as YouTube or youtube or YOUTUBE. You can find the URLs easier by converting the string to lowercase. However, you will need the original string (with uppercase and lowercase letters) to view the YouTube URL to answer a quiz question because YouTube links are case sensitive. The link https://www.youtube.com/watch?v=ji5_MqicxSo is different than the link https://www.youtube.com/watch?v=ji5_mqicxso, where all the letters are lowercase.