You can find a PDF of this programming exercise in the **Resources** tab.

For files related to this assignment, visit the DukeLearnToProgram Project Resources page for this course: http://www.dukelearntoprogram.com/course2/files.php (also linked in the **Resources** tab).

You can also find the frequently asked questions page for this course's assignments on DukeLearnToProgram: http://www.dukelearntoprogram.com/course2/faq.php (also linked in the **Resources** tab).

## Part 1: Finding many Genes

This assignment is to write the code from the lesson to make the following improvements to your algorithm:

A. Find a gene in a strand of DNA where the stop codon could be any of the three stop codons "TAA", "TAG", or "TGA".

B. Find all the genes (where the stop codon could be any of the three stop codons) in a strand of DNA.

This will help you see if you really understood how to put the code together, and might identify a part that you did not fully understand. If you get stuck, then you can go back and watch the coding videos that go with this lesson again.

Specifically, you should do the following:

1. Create a new Java project named StringsSecondAssignments. You can put all the classes for this programming exercise in this project.

2. Create a new Java Class named Part1. The following methods go in this class.

3. Write the method findStopCodon that has three parameters, a String parameter named dna, an integer parameter named startIndex that represents where the first occurrence of ATG occurs in dna, and a String parameter named stopCodon. This method returns the index of the first occurrence of stopCodon that appears past startIndex and is a multiple of 3 away from startIndex. If there is no such stopCodon, this method returns the length of the dna strand.

4. Write the void method testFindStopCodon that calls the method findStopCodon with several examples and prints out the results to check if findStopCodon is working correctly. Think about what types of examples you should check. For example, you may want to check some strings of DNA that have genes and some that do not. What other examples should you check?

5. Write the method findGene that has one String parameter dna, representing a string of DNA. In this method you should do the following:

- Find the index of the first occurrence of the start codon “ATG”. If there is no “ATG”, return the empty string.
- Find the index of the first occurrence of the stop codon “TAA” after the first occurrence of “ATG” that is a multiple of three away from the “ATG”. Hint: call findStopCodon.
- Find the index of the first occurrence of the stop codon “TAG” after the first occurrence of “ATG” that is a multiple of three away from the “ATG”. Find the index of the first occurrence of the stop codon “TGA” after the first occurrence of “ATG” that is a multiple of three away from the “ATG”.
- Return the gene formed from the “ATG” and the closest stop codon that is a multiple of three away. If there is no valid stop codon and therefore no gene, return the empty string.

6. Write the void method testFindGene that has no parameters. You should create five DNA strings. The strings should have specific test cases such as DNA with no “ATG”, DNA with “ATG” and one valid stop codon, DNA with “ATG” and multiple valid stop codons, DNA with “ATG” and no valid stop codons. Think carefully about what would be good examples to test. For each DNA string you should:

- Print the DNA string.
- Calculate the gene by sending this DNA string as an argument to findGene. If a gene exists following our algorithm above, then print the gene, otherwise print the empty string.

7. Write the void method printAllGenes that has one String parameter dna, representing a string of DNA. In this method you should repeatedly find genes and print each one until there are no more genes. Hint: remember you learned a while(true) loop and break.

## Part 2: HowMany - Finding Multiple Occurrences

This assignment will write a method to determine how many occurrences of a string appear in another string.

Specifically, you should do the following:

1. Create a new Java Class named Part2in the StringsSecondAssignments project.

2. Write the method named howMany that has two String parameters named stringa and stringb. This method returns an integer indicating how many times stringa appears in stringb, where each occurrence of stringa must not overlap with another occurrence of it. For example, the call howMany(“GAA”, “ATGAACGAATTGAATC”) returns 3 as GAA occurs 3 times. The call howMany(“AA”, “ATAAAA”) returns 2. Note that the AA’s found cannot overlap.

3.Write the void method testHowMany has no parameters. Add code in here to call howMany with several examples and print the results. Think carefully about what types of examples would be good to test to make sure your method works correctly.

## Part 3: How Many Genes?

Write a program to count how many genes are in a strand of DNA.

Specifically, you should do the following:

1. Create a new Java Class named Part3 in the StringsSecondAssignments project. Put the following methods in this class.

2. Copy your methods from Part1 to find one gene and print all genes.

3. Write the method named countGenes that has a String parameter named dna representing a string of DNA. This method returns the number of genes found in dna. For example the call countGenes("ATGTAAGATGCCCTAGT") returns 2, finding the gene ATGTAA first and then the gene ATGCCCTAG. Hint: This is very similar to finding all genes and printing them, except that instead of printing all the genes you will count them.

4. Write the void method named testCountGenes that has no parameters. This method calls countGenes with many example strings and prints the result for each. You should create several examples with different numbers of genes to test your code.

Mark as completed