Assignment 7

Overview

This assignment is similar to Assignment 6 in that it will process a DNA string, but it is slightly more complex. Here, the program will be given two arguments on the command line. One will be the name of a file containing lines that identify *restriction enzymes*, and the second will be a file containing a DNA string. For every line in the file containing the restriction enzymes, it will apply the information in that line to the DNA contained in the DNA file. Roughly, a restriction enzyme acts like a pair of scissors and a template. It is a pattern that specifies a place in the DNA string at which to cut the DNA. It is explained in more detail below.

The details of the assignment are specified in the *Detailed Requirements* section below. The *Background* section that follows is a short tutorial on the subject of restriction enzymes. It includes some background from previous assignments so that this assignment is self-contained. This assignment will give you experience in reading from files and in more sophisticated pattern-matching.

Background

DNA and Nucleotides

A *DNA string*, also called a *DNA strand,* is a finite sequence consisting of the four letters `A`, `C`, `G`, and `T` in any order[1]. The four letters stand for the four *nucleotides*: *adenine*, *cytosine*, *guanine*, and *thymine*. In this assignment, the nucleotides will always be in *uppercase* format. Nucleotides, which are the molecular units from which DNA and RNA are composed, are also called *bases*. Each nucleotide has a *complement* among the four: `A` and `T` are complements, and `C` and `G` are complements. Complements are chemically-related in that when they are close to each other, they form hydrogen bonds between them.

In its most common form, DNA is actually a double helix consisting of two strands that wrap around each other. Each DNA strand has direction. Direction is usually indicated by putting a 5' at one end and a 3' at the other. The 5' and 3' refer to the names of the carbon atoms to which these ends attach. For example,

```
5'-GTATCC-3'
```

is a fragment of DNA that runs from the 5' to the 3' position. The 5' end is called the *upstream* end, and the 3' end is the *downstream* end. The two strands of nucleotides are in reverse directions of each other. In other words, if you could unwind the helix so that the two strands were lying on a flat surface parallel to each other, in one strand the 5' end would be to the left, and in the other, it would be to the right. The two strands are chemically-related because the bases that would be across from each other on the table are complements of each other. For example, the two strands below

```
5'-G T A T C C A A T G C C-3'
   | | | | | | | | | | | |
3'-C A T A G G T T A C G G-5'
```

could be a fragment of the unwound double helix. The vertical lines connect complements in the forward and reverse strands to each other. Each `C` in one is matched by a `G` in the other, and each `A` is matched by a `T` in the other.

[1] It does not matter whether they are in uppercase or lowercase.

Restriction Enzymes

Bacteria produce special enzymes that can cut their DNA at specified sites, called *cleavage sites*. The cleavage site is a position between two nucleotides in the DNA. The enzyme finds its site by a type of biological pattern-matching. The pattern specifies where in the DNA the enzyme will match. The place on the DNA molecule that matches the pattern is called the *recognition site*. For example, the enzyme *EcoRI* has a recognition site defined by

```
5'-G'AATTC-3'
```

The apostrophe ' between the G and the A is the cleavage site. This means that *EcoRI* will search for a substring of the DNA consisting of the bases GAATTC in the 5' to 3' direction, and cut the DNA between the G and the first A. So, if the DNA string is as below (with the parts that match the pattern, i.e., the recognition sites, in **bold**)

```
ATGAAAGGGTTTCCCTTTGAATTCCCCATGGTATTGTTGCCGGAATTCTTTCCGGCCCCC
```

it will be cut into the three pieces

```
ATGAAAGGGTTTCCCTTTG     AATTCCCCATGGTATTGTTGCCGG     AATTCTTTCCGGCCCCC
```

by *EcoRI*. The restriction enzyme *NotI* is defined by

```
5'-GC'GGCCGC-3'
```

which indicates that it will find all occurrences of the string GCGGCCGC in the 5' to 3' direction and cut the DNA between the first C and the second G.

You may have noticed that if you form the complement of GAATTC, you get CTTAAG, which is the string spelled backwards. Similarly, the complement of GCGGCCGC is CGCCGGCG, which is also the string spelled backwards. Certain types of restriction enzymes have this *palindromic* property.

Some restriction enzymes have a cleavage site outside of the recognition site. *AceIII* is defined by

```
CAGCTCNNNNNNN'
```

The N is a symbol that matches any of A, C, G, or T. Therefore, this enzyme cuts the DNA between the 7th and 8th nucleotides after its recognition site. For example, the short fragment of DNA

```
CAGCTCAAATGCCAGGGGGGG
```

will be cut between the C and the A:

```
CAGCTCAAATGCC  AGGGGGGG
```

In actuality, many of these restriction enzymes cleave both strands of the DNA at once, and not necessarily at the same position. We are going to simplify the problem in this assignment and assume that the DNA is single-stranded and that it is cut as described above.

Detailed Requirements

The program must be run with *two* command line arguments. Suppose the program is named `cleave`. Then the command

```
cleave enzyme_file mydna_file
```

will cause the program to read the enzyme_file, and for each enzyme in the file, to apply that enzyme's cleaving to the DNA in the file named `mydna_file`. Applying an enzyme to the DNA will result in a set of smaller fragments of DNA, which will be placed in an output file. There will be one fragment per line in the output file, arranged in the order in which they occur in the input file. For example, if the DNA is cut at positions 20, 175, 300, and 350, there will be five fragments, consisting of the DNA from 1 to 20, 21 to 175, 176 to 299, 300 to 350, and 351 to the end. Here the position is the base after which the cut occurs.

The name of the output file is constructed by appending the name of the restriction enzyme to the name of the DNA file, with an underscore in between them. For example, if the enzyme is *EcoRI* and the DNA file is named *BC161026*, the output file should be named *BC161026_EcoRI*. This implies that the number of output files is equal to the number of restriction enzymes in the enzyme_file.

The enzyme file will be a simplified version of a format known as the *Staden* file format. Each line in the file has the form

```
enzyme_acronym/recognition_sequence//
```

The cut point will be denoted by an apostrophe in the recognition sequence. An example of an actual Staden enzyme file is:

```
AatI/AGG'CCT//
AatII/GACGT'C//
AbsI/CC'TCGAGG//
AccII/CG'CG//
AccIII/T'CCGGA//
Acc16I/TGC'GCA//
```

The first line is the enzyme named *AatI* and its cut point is between the second `G` and the first `C`. Notice that none of the enzymes above have the `N` symbol. In an actual Staden file, there are many symbols besides `A`, `C`, `G`, and `T` in the recognition sequences. These letters are part of a standard set of abbreviations defined as follows:

Listing 1: Staden File Format Specifiers

```
R = G or A
Y = C or T
M = A or C
K = G or T
S = G or C
W = A or T
B = not A (C or G or T)
D = not C (A or G or T)
H = not G (A or C or T)
V = not T (A or C or G)
N = A or C or G or T
```

To simplify this assignment, the only letters that you will find in the enzyme recognition sequences are `A`, `C`, `G`, ***and*** `T`. *But you should be aware that this is a very big simplification of what really happens.*

The DNA file may contain multiple lines. For example, it could look like

```
GGCTCATCATGATGCGCGCAGTTCTTCTATTCGGCTGTTCCCTATTAATCGTCGCCAGGGCCAATATTCC
CGAGGAGCGGGATGTACTGGTGCTAAAGAAAGACAACTTTGATGAGGCGTTAAAGCAGTACCCGTTTATT
CTAGTGGAATTCTATGCTCCCTGGTGTGGTCACTGCAAGGCACTGGCTCCAGAATATGAAAAGGCTGCTG
GCGTATTAAAAAGTGAAGGTTTGCCGATCCGCCTGGGCAAGGTAGATGCTACAGAGGAGTCTGATCTGGC
...
```

The program must ignore the newline characters and treat the data as if it were one long string. (Hint – how can you read the lines and then glue them into a single string?)

Error Checking

If either file on the command line cannot be opened (because it does not exist, or the program does not have permission to open it), the program should display a specific message related to the error and then exit. If the DNA file contains any symbols other than `A`, `C`, `G`,`T`, and newline characters, the program should exit with an appropriate error message. Note that the input DNA must be in uppercase only. For simplicity, the program can assume that the enzyme file is in the proper format and contains no spaces. The program does not have to check that the enzyme file is in the wrong format.

Program Considerations

Because this program involves several different tasks, and because time is limited, this section tells you the large chunks that you need to consider creating.

1. Check that the command line arguments are correct and valid and handle the errors.
2. Open the enzyme file, and from each line, extract the following information: the name of the enzyme, the pattern, and the distance from the start of the pattern to the cleavage site. How will you store this information? For each enzyme name there are two pieces of associated information: the pattern and the cleavage position.
3. Open and read the DNA file, and somehow create a single DNA string from it, with nothing but the base letters.
4. For each enzyme, create an output file with the proper name, and then apply the enzyme to the DNA, cutting it into pieces, and putting each piece on a separate lien in the output file. Close the output file when you are finished with it!

The hardest part is cleaving the DNA. As a reminder we learned about Perl's special variables, $‘, $&, and $’. One or more of them might be useful. Equally important, you will need the substr() function. Read about it. Remember that after the DNA is cleaved, the enzyme will start looking for a match in the part immediately after the cleavage site. As an extreme example, suppose we had an enzyme with the recognition sequence `AA'AAA` and the DNA string was

```
AAAAAAAAAAAAAAA
```

Then it will be cleaved first into

```
AA AAAAAAAAAAAA
```

and again into

```
AA AA AAAAAAAAAA
```

and so on. The point is that after the enzyme is applied, the DNA string to be matched is the one starting right after the cleavage site.

Finally, as a reminder, the program must have a proper preamble containing the program name, authorship, date of creation, usage information, a description of the input that it accepts and what it does when given that input.

The output can contain nothing but the DNA fragments – no messages to the user, no extra lines, no spaces. The output might be needed again as the input to a different program, and it is important that it is in the format that I described above.

Testing Your Program

All programs must be thoroughly tested before they are released to users. Create some sample input files of a small size and manually figure out what the outputs should be. Run your program and make sure that your output matches the one you manually computed. There are a few DNA files in the directory `/data/biocs/b/student.accounts/cs132/data/nucleotides`. Each is in three formats: a single line with no newlines, FASTA format, and a text format with newlines. More files can be created using the Perl script in the demos directory that makes random DNA strings. In addition, you can download DNA files from the GenBank or other such archives.

Grading Rubric

The grade on the program is based on its correctness (80%), its style (10%), and its documentation (10%).

Extra Credit

For 25% extra credit, if you have the time and are interested, make the program work with arbitrary Staden formats shown in Listing 1.

Submitting the Solution

Place your completed project in `/data/yoda/b/student.accounts/cs132/projects/project7` and name it `hwk7_`*`username`*`.pl`. Make sure that your file is readable only by you (permission 600). Remember, this is your work, and your work alone.