Homework #3
Biocomputing 1: Perl for bioinformatics (140.636)

Preliminaries

- Suggested reading:
  - Regular expressions: Schwartz, Chapters 7, 8, 9
  - Intermediate Perl programming: 10, 11, 15
  - Intermediate programming lecture notes
  - Processing command lines: CommandProcessing.pdf

- Remember you may consult with fellow students and discuss the homework, say by working out algorithms and data structures on a blackboard, but you must write your own code and it is not permitted to exchange code fragments.

- Your perl script should be called **`resmap.pl`**

Background

You will write a program that performs *in silico* restriction enzyme digests. We will use the program to generate a simulated gel electrophoresis analysis of *Bacteriophage Lambda*. The DNA of this phage is linear. This background is taken from:

http://biotech.biology.arizona.edu/labs/DNA_analysis_RE_student.html).

Restriction enzymes (or restriction endonucleases) cleave DNA in a very specific fashion. Type II restriction enzymes, most commonly used for DNA analysis and genetic engineering, each have a unique nucleotide sequence at which it cuts a DNA molecule. A particular restriction enzyme will cleave DNA at that recognition sequence and nowhere else. The recognition sequence is often a six base pair *palindromic* sequence (the top DNA strand from 5' to 3' is the same as the bottom DNA strand from 5' to 3'), but others recognize four or even eight base pair sequences.

Restriction enzymes can also differ in the way they cut the DNA molecule. Some enzymes cut in the middle of the recognition sequence, resulting in a flush or blunt end. Other enzymes cleave in a staggered fashion, resulting in DNA products that have short single-stranded overhangs (usually two or four nucleotides) at each end. These are often called cohesive ends.

Enzyme: *Alu I*
Recognition sequence and cleavage site: AG|CT

```
N-N-A-G-C-T-N-N              N-N-A-G         C-T-N-N
| | | | | | | |       →      | | | |    +    | | | |
N-N-T-C-G-A-N-N     Alu I    N-N-T-C         G-A-N-N
```

Enzyme: *Eco RI*
Recognition sequence and cleavage site: G|AATTC

```
N-N-G-A-A-T-T-C-N-N          N-N-G                  A-A-T-T-C-N-N
| | | | | | | | | |    →     | | |           +              | | |
N-N-C-T-T-A-A-G-N-N  Eco RI  N-N-C-T-T-A-A                  G-N-N
```

A common use for restriction enzymes is to generate a "fingerprint" of a particular DNA molecule. Because of the sequence specificity of restriction enzymes, these enzymes can cut DNA into discrete fragments that can be resolved by gel electrophoresis. This pattern of DNA fragments generates a "DNA fingerprint." Each DNA molecule has its own fingerprint. Different restriction enzymes produce different fingerprints. The location of these restriction enzyme cleavage sites on the DNA molecule can be compiled to create a *restriction enzyme map*. These maps are useful in cloning for identifying and characterizing a particular DNA plasmid or region.

Problem

The script you will write will be named **`resmap.pl`** and will take as its input a DNA sequence and the name of a restriction enzyme as well as an option indicating whether the DNA is linear or circular. The input DNA sequence should be input via standard input. The enzyme should be input as a parameter on the command line. For example, if the sequence is in the file **`theSequence.seq`** and the desired enzyme is **`EcoRI`** , then command line would look like this:

```
> resmap.pl –e EcoRI   theSequence.seq
```

If the DNA is circular then the command line should look like this:

```
> resmap.pl –c –e EcoRI   theSequence.seq
```

The **`–c`** command line option tells your script that the input sequence should be treated as circular DNA. Otherwise it should treat the DNA sequence as a linear DNA sequence. Whatever follows the **`-e`** option is interpreted as the name of the restriction enzyme. It will use this name to lookup the properties of the enzyme in order to perform the desired digests.

The restriction enzymes will be applied to two DNA sequences that are in

```
/users/sph140636/shared/homework3
```

1) The file **`NC_001416.seq`** contains the linear DNA sequence of *Bacteriophage lambda*.
2) The file **`NC_001764.seq`** contains the circular DNA sequence of the *Bacillis Subtilis* pTA1040 plasmid.

Your script should have the following functionality

1. Print help information if there are no command line arguments or missing command line arguments.
2. Read the enzyme characteristics from enzymes.txt and load them into a hash (see hits below)
3. Get the enzyme name from the command line. It should appear AFTER the **`–e`** option flag (see hints below)
4. Get the sequence <u>from standard input</u>
5. Read the sequence into a single string variable
6. Scan the sequence for the recognition sequences of the selected enzyme and generate the appropriate fragments (take care with circular DNA!)
7. Calculate and print the lengths of each of the resulting restriction fragments.
8. Print the <u>sorted</u> fragment lengths.

Your script should provide a help message. In other words, if a naïve user simply types the name of script, without arguments, the script will tell the user how to run the script.

In addition to the usual problem description, your solution web page should contain an extremely well documented script and six sets of results (corresponding to the two DNA sequences and the three endonucleases). The instructor will run your script to verify that it works as advertised.

Hints

(1) There are many ways of reading the input file and of processing command line options. The easiest way to read the file, is with the diamond operator (pp. 88-90 of "Learning Perl"). The diamond operator extends the functionality of <STDIN>. The easiest way of processing the options is with the **`Getopt`** module. Read commandLineProcessing.pdf (in `/users/sph140636/shared/notes_handouts/`) for more details. Also take a look at:.

http://www.cs.mcgill.ca/~abatko/computers/programming/perl/howto/getopts/

(2) Resttriction enzyme characteristics

The file **`re.pl`** contains a hash with characteristics of many restriction enzymes. You can paste the hash directly into your Perl script. Each entry in the hash is an anonymous array with the pattern and the position of the cut. For example, the comment after each entry below shows the location of the cut.

```
%re2 = (
      'BamHI'   => ['GGATCC',1],       # G | GATCC
      'EcoRI'   => ['GAATTC',1],       # G | AATTC
      'EcoRII'  => ['CCWGG', 0]       #    | CCWGG
);
```

(3) To locate recognition sites, you will need to do global pattern matching. You will need the **`pos()`** function to get the position information from the global pattern matching. Here is an example of using the **`pos()`** function:

```
while($string =~ /pattern/g) {
      print pos($string),"\n";
}
```

(4) To sort the fragment lengths you will have to sort numerically. Here is how to make the sort function sort in numerical order rather than lexical order (see p. 213 of Schwartz).

```
@result = sort {$a ⇔ $b} @unsorted;
```

(5) Be careful with the indexing. It's easy to get it wrong. Try testing your program with a small sequence of your own creation where you can easily check that the fragment sizes are correct. Check the results for linear and circular DNA! You can check that your code is cutting in the right place at this web site: `http://nc2.neb.com/NEBcutter2/`

(6) Here is what happens if I run my script without arguments

```
> resfrag.pl

PURPOSE:        generates restriction fragments
USAGE:          resfrag.pl -e [enzyme] < [sequencefile]
                sequence file contains sequence only, no header
SUPPORTED ENZYMES:
                EcoRII
                BamHI
                EcoRI
```

(7) Here is what happens if I run my script on the *Bacteriophage lambda* DNA sequence:

```
resfrag.pl -e EcoRI < NC_001416.seq
INPUT SEQUENCE LENGTH:48502
RESTRICTION ENZYME   :EcoRI
RESTRICTION FRAGMENTS (FOR LINEAR DNA):
        3.53 kb
        4.878 kb
        5.643 kb
        5.804 kb
        7.421 kb
        21.226 kb
CHECK CUMMULATIVE FRAGMENT LENGTH: 48502
```