Problems for Introduction to Perl

Processing strings

One of the most common activities in bioinformatics is to take large strings, such as nucleotide sequences, and search through them, dice them up, manipulate them in a variety of ways. Since this is all difficult to do by hand, a basic skill is to use computer programming to manipulate strings. Here's an example:

1. You've downloaded a DNA sequence in FastA format.[1] The file contains the sequence of the gene you've been seeking for weeks. Now it's here,… unfortunately the gene is flanked by upstream and downstream sequence. You know the starting and ending coordinates, but who wants to count hundreds of nucleotides (and possibly make a mistake). This is something that computers do better.

   As luck would have it, you know of a Perl program designed to extract DNA sequences from larger sequences. It's called `Get_gene.pl`. You can get both the program and the file on the course calendar. Just run the program and get your gene? … It's seldom that easy.

   Modify the program so that it picks out the gene that you've been given to believe begins at coordinate 97 and ends at coordinate 977. Make sure you verify that the gene extracted is reasonable. Have the program print out just the gene.

   Hint: *Use the* `substr( )` *function*, *which extracts an internal string from a larger one. Here's the syntax (or try going to one of the Perl resources accessible from the Links page on the course's web site):*

   ```
   $string = substr($originalString, $offset, $length);
   ```

   OR

   ```
   $string = substr($originalString, $offset);
   ```

   The $offset argument is the start of the substring you are interested in, counting from the front (if its positive) or from the end (if its negative). If $offset == 0, then the substring starts at the beginning.

   $length is the length of the substring, if omitted, substring will be from $offset to the end of $originalString.

   Examples:

   ```
   print substr("Have a nice day!", 7, 4);   gives   nice
   print substr("Have a nice day!", 7);      gives   nice day!
   ```

---

[1] FastA format is one of the most commonly encountered ways that nucleotide and protein sequences are formatted. It consists of a single header line identified by the character ">" in the first position and any number of subsequent data lines. The header lines can contain anything at all – usually things like the name of the gene, the name of the organism, etc. The data lines consists of nothing except nucleotide or amino acid sequences, for example GGAGCTTG. Here's a brief example:

```
>Rabbit alpha-globin cDNA  Accession J00658   552 bp
ACACTTCTGGTCCAGTCCGACTGAGAAGGAACCACCATGGTGCTGTCTCC
CGCTGACAAGACCAACATCAAGACTGCCTGGGAAAAGATCGGCAGCCACG
[and so forth]
```

2. Learn something about the sequence you've been given. Modify the program so that it prints out the number of nucleotides in the file before the gene begins, then the number of nucleotides within the gene, then the number of nucleotides after the gene.

   Hint: *Use the* `length( )` *function, which gives you the length of the referenced string. For example:*

   ```
   print length("Have a nice day!");          gives   16
   ```

3. The sequence in the file is lower case, and you ***hate*** DNA sequences printed in lower case. So modify the program so that it prints the gene in upper case.

4. You'll want to identify specific nucleotides in your sequence, and you still want to avoid needless counting. Modify the program so that it prints out the gene in chunks of 50 nucleotides to facilitate counting.

5. Genes from a given organism generally have the same (G+C):(A+T) ratio, more or less. It is therefore useful to know what that ratio is for a gene. For example, you can use it to assess the likelihood that a gene was recently acquired by the organism in evolutionary time. Write a loop that will examine each nucleotide of the gene and count (G+C) and (A+T).

6. Since DNA is double stranded and DNA sequences really are TWO sequences, one for each strand. However, since the two strands are complementary to each other, it's considered a waste of space to print both or to store both in a file. This sometimes causes problems. Suppose you're given a DNA sequence (one of the two strands, of course), and unfortunately, the gene you want is on the OTHER strand. Modify the program so that it will produce the sequence of the other strand (if you know something about molecular biology, recall that DNA is antiparallel, i.e. if you read one strand left-to-right, you need to read the other right-to-left – and complementary) (if you <u>don't</u> know something about molecular biology, patience. You will next week).