Assignment 6

Overview

In this assignment, your program will simulate protein synthesis from DNA. Specifically, it reads a DNA string from a file specified on the command line, synthesizes the protein that would be created from this DNA string by the human ribosome, and writes that protein's representation to the standard output stream. 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 protein synthesis. This assignment will give you experience in using hashes and in processing text using Perl's pattern-matching capabilities.

Background

A *DNA string*, also called a *DNA strand,* is a finite sequence consisting of the four lowercase letters `a`, `c`, `g`, and `t` in any order. The four letters stand for the four *nucleotides*: *adenine*, *cytosine*, *guanine*, and *thymine.* Nucleotides, which are the molecular units from which DNA and RNA are composed, are also called *bases.* A special enzyme called *RNA polymerase* uses the information in DNA to create RNA. A *RNA string* or *RNA strand* is a finite sequence consisting of the four lowercase letters `a`, `c`, `g`, and `u`. The `a`, `c`, and `g` have the same names as they do in DNA, but the `u` represents *uracil.* When DNA is transcribed to RNA by RNA polymerase, each thymine base is converted to uracil. Hence RNA strings have `u`'s wherever DNA has `t`'s.

RNA in turn serves as a template for the construction of *proteins*, which are sequences of *amino acids.* Proteins are synthesized within the *ribosomes* of living cells by a process called *translation.* In translation, the RNA string is read in three-letter groups called *codons.* Each codon codes for a particular amino acid. For example, `guu` codes for *valine*, and `uca` codes for *cysteine.* Let us count how many possible three-letter sequences there are in which each letter can be `a`, `c`, `g`, or `t`. There are four choices for the first letter, four independent choices for the second letter, and four for the third, so there are $4^3 = 64$ different codons. On the other hand, there are only 20 different amino acids. Some amino acids are coded for by multiple codons. For example, `uca`, `ucc`, `ucg`, and `ucu` all code for *cysteine.*

Some codons do not code for any amino acids; they are *stop codons.* Their purpose is to temporarily terminate protein synthesis while reading the RNA string. There are three such stop codons: `uaa`, `uag`, and `uga`. When a stop codon is reached in the RNA string, protein synthesis is paused. The RNA continues to be read, but amino acids are not created until it sees a special *start codon.* When it finds a start codon, translation begins again. This is very much like the way that the Perl compiler treats the comments in your Perl programs. As it reads your program, it translates it into executable instructions until it finds a comment start character #. The `#` tells the compiler to stop compiling, and the next newline character tells it to start compiling again. The `#` acts like a stop code and the newline, like a start code. The start codon in RNA is `aug`, which also codes for *methionine* (Met).

Thus, not all of a RNA string is translated into protein; there are large regions that are gaps in the translation process. As the RNA is read, when a gap is reached, it is skipped over until a start codon is found. These gaps are sometimes much larger than the non-gaps in the RNA. As the ribosome reads the RNA, it splices together the separate pieces that it has translated. As an example, the RNA strand

```
augguuuauggucucuga
```

is read as the following sequence of codons

```
aug guu uau ggu cuc uga
```

Consulting the tables on page 3 of this assignment, we see that `aug` is a start codon that codes for *methionine* (Met), `guu`, for *valine* (Val), `uau`, for *tyrosine* (Tyr), `ggu`, for *glycine* (Gly), `cuc`, for *leucine* (Leu), and `uga` is a stop codon. Therefore, the sequence *Met-Val-Tyr-Gly-Leu* is created from this RNA fragment.

Amino acids have long names like cysteine but they also have three-letter names such as Cys, for cysteine, and one-letter uppercase names, such as C for cysteine. It is not always true that the one-letter name is the first letter of the amino acid's long name. The above sequence would be written *MVYGL* using the one-letter names. The table on page 3 contains the one-letter names as well.

Detailed Requirements

The program must use the file named on the command line as its input file. Suppose the program were named `translate`. Then if the user types

```
translate mydna_file
```

the program must read the data in `mydna_file`. If the input file contains any symbols other than `a`, `c`, `g`, or `t`, the program should exit with an appropriate error message. The file can be assumed to contain no newlines or white-space characters; the program does not have to check that this is true. (It will be a harder program to write if it is not true.)

The program should read the input file, translate it into RNA, and then transcribe the RNA into a sequence of uppercase one-letter amino acid names.

Translation must always start at the very first character in the file. Whenever it finds a gap in the RNA, it should start a new line of output. For example, if the RNA string is of the form *<seq1>gap<seq2>gap<seq3>*, then the proteins coded by *seq1*, *seq2*, and *seq3* will be on separate lines. The output should contain nothing else, no blank lines, no sentences – just the amino acid sequences, one per line.

For simplicity, the program can assume that the lengths of the gaps are always multiples of three characters. This is not always the case in reality, but it will greatly simplify the logic within the program. As an example, the input file may have a sequence of length 90 followed by a gap of length 120, followed by a sequence of length 42 and a gap of length 96 and end in a sequence of length 240. The actual input files will be much larger than this.

Program Considerations

Hashes are an ideal data structure for this program. If your program does not use a hash, it is not designed properly. We have not yet learned about functions, but if you want to, you can use functions to simplify your main program. This is not a requirement.

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 program must be self-contained; it cannot depend upon the existence of a data file. When you think through how this program should be designed, you will realize that in order for it to translate the RNA to single-letter amino acid names, it will have to store that mapping somewhere. It must be within the program file itself.

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.

Submitting the Solution

Place your completed project in `/data/biocs/b/student.accounts/cs132/projects/hwk6` and name it `hwk6_`*`username`*.pl. Make sure that your file is readable only by you. Make sure that each script has a proper prologue in the beginning (after the first line specifying the Perl interpreter) and that it is documented well. If you do not remember what makes a program good, consult the summary slide of Lesson 12. Finally, remember that this is your work, and your work alone.

Codon Tables

Codon	Amino Acid	Codon	Amino Acid	Codon	Amino Acid	Codon	Amino Acid
uuu	Phe	ucu	Ser	uau	Tyr	ugu	Cys
uuc	Phe	ucc	Ser	uac	Tyr	ugc	Cys
uua	Leu	uca	Ser	uaa	TER	uga	TER
uug	Leu	ucg	Ser	uag	TER	ugg	Trp
cuu	Leu	ccu	Pro	cau	His	cgu	Arg
cuc	Leu	ccc	Pro	cac	His	cgc	Arg
cua	Leu	cca	Pro	caa	Gln	cga	Arg
cug	Leu	ccg	Pro	cag	Gln	cgg	Arg
auu	Ile	acu	Thr	aau	Asn	agu	Ser
auc	Ile	acc	Thr	aac	Asn	agc	Ser
aua	Ile	aca	Thr	aaa	Lys	aga	Arg
aug	Met	acg	Thr	aag	Lys	agg	Arg
guu	Val	gcu	Ala	gau	Asp	ggu	Gly
guc	Val	gcc	Ala	gac	Asp	ggc	Gly
gua	Val	gca	Ala	gaa	Glu	gga	Gly
gug	Val	gcg	Ala	gag	Glu	ggg	Gly

Three-letter name	One-letter name
Ser	S
Phe	F
Leu	L
Tyr	Y
Cys	C
Trp	W
Pro	P
His	H
Gln	Q
Arg	R
Ile	I
Met	M
Thr	T
Asn	N
Lys	K
Val	V
Ala	A
Asp	D
Glu	E
Gly	G