Introduction to Perl Programming
– Regular Expressions/String Manipulation

Beginning Perl, Chap 5

Previous Exercises

1. Take a set of FASTA format files and make a FASTA library, combining those files
2. Take a FASTA library, and break it into individual FASTA files
3. Write a perl equivalent to "`mrtrans`" that does not care about the order of sequence input. Given two files, a FASTA protein library (with '-' for gaps) and a FASTA DNA library with the same sequence names, generate a FASTA DNA alignment with '---' inserted into the DNA sequence for each '-' in the protein sequence

Exercise 1 - to_fasta.pl

```
#!/usr/bin/perl -w
# to_fasta.pl - concatenate a set of files into a fasta library
#
use strict;

use vars qw($file $line);

foreach $file ( @ARGV ) {
  if (!open(INFILE,"< $file")) {
    warn "cannot open $file";
    next;
  }
  $line = <INFILE>;
  if (substr($line,0,1) ne '>') { print ">$file\n";}
  print $line;
  while ($line = <INFILE>) { print $line;}
  close(INFILE);
}
```

Exercise 2 – from_fasta.pl

```
#!/usr/bin/perl -w
use strict;
use vars qw($ofile $line $f_cnt);

open(INFILE,"< $ARGV[0]") || die(" could not open $ARGV[0]");

$ofile = ""; $f_cnt = 0;
while ($line = <INFILE> ) {
  if (substr($line,0,1) eq '>') {
    if ($ofile) {close (OUTFILE);}
    $ofile = "file".$f_cnt;
    $f_cnt += 1;    # $f_cnt++;
    open(OUTFILE,">$ofile");
  }
  print OUTFILE $line;
}
close OUTFILE;
close INFILE;
```

Exercise 2 - advanced to_fasta.pl

```
#!/usr/bin/perl -w
use strict;
use vars qw($num $entry);

$num = 0;
{ local $/ = "\n>";   # change the line separator to "\n"
  while ($entry = <INFILE>) {
    chomp($entry);
    $entry =~ s/\A>?/>/; # replace missing >, if necessary
              # \A is like ^ (beginning of string)
    open(OUT, ">file$num.fa") or die $!;
    $num++;
    print OUT "$entry\n";
    close(OUT);
  }
}
```

Regular expressions

```
>gi|121694|sp|P20432|GTT1_DROME Glutathione S-transferase 1-1
```

- used for string matching, substitution, pattern extraction
- `/^>gi\|/` matches `>gi|121694|sp|P20432|GTT1_DROME …`
- `if ($line =~ m/^>gi/) { … } #match`
- `$line =~ /^>gi\|(\d+)\|/; # extract gi#`
  `$gi = $1;`
- `($gi) = $line =~ /^>gi\|(\d+)\|/; #same`
- `$line =~ s/^>(.*)$/>>$1/; # substitution`

Regular expressions (cont.)

```
>gi|121694|sp|P20432|GTT1_DROME Glutathione S-transferase 1-1
```

- m/plaintext/
  m/one|two/; # alternation
  m/(one|two)|(three)/; # grouping with
  # parenthesis
- /^>gi\|(\d+)/ #beginning of line
  /.+ (\d+) aa$/ # end of line
- /a*bc/ # bc,abc,aabc, … # repetitions
  /a?bc/ # abc, bc
  /a+bc/ # abc, aabc, …

Regular Expressions, III

```
>gi|121694|sp|P20432|GTT1_DROME Glutathione S-transferase 1-1
```

- Matching classes:
  - /^>gi\|[0-9]+\|[a-z]+\|[A-Z][0-9a-z]+\|/
    - [a-z] [0-9] -> class
    - [^a-z] -> negated class
  - /^>gi\|\d+\|[a-z]\|\w+\|/
    - \d -> number [0-9] \D -> not a number
    - \w -> word [0-9A-Za-z_] \W -> not a word char
    - \s -> space [\t\n\r] \S -> not a space
- Capturing matches:
  - /^>gi\|(\d+)\|([a-z])\|(\w+)\|/
    $1 $2 $3
  - ($gi,$db,$db_acc) =
    $line =~ /^>gi\|(\d+)\|([a-z])\|(\w+)\|/;

Regular expressions - modifiers

- `m/That/i` # ignore case
- `s/this/that/g` # global replacement
- `m/>gi\|(\d+)\|([a-z]{2,3}|(\w+) #{range}`
- `s///m` # treat string as multiple lines
- `s///s` # span over \n
- `s/\n//gs` # remove \n in multiline entry
- `s/GAATTC/$1\n/g` # break lines at EcoRI site

```
{ local $/ = "\n>";   # change the line separator to "\n"
  while ($entry = <INFILE>) {
    chomp($entry);
    $entry =~ s/\A>?/>/; # replace missing >, if necessary
              # \A is like ^ (beginning of string)
    open(OUT, ">file$num.fa") or die $!;
    $num++;
    print OUT "$entry\n";
    close(OUT);
  }
```

String expressions (with regular expressions)

- `if ( /^>gi\|/ ) { … }`
- `if ( $line =~ m/^>gi\|/) { … }`
- `while ( $line !~ m/^>gi\|/) { … }`
- Substitution:
  `$new_line =~ s/\|/:/g;`
- Pattern extraction:

```
($gi,$db,$db_acc) =
    $line = /^>gi\|(\d+)\|([a-z])\|(\w+)\|/;
```

- **split** (/|/,$line)
- **join** (“:”,@fields);
- substr($string,$start,$length); # rarely used
- index($string,$query); # rarely used
- Comparison: “eq” “ne” “cmp” “lt” “gt” “le” “ge”

Perl mrtrans.pl

```
use vars qw(%proteins %dnas
        $line $name $codon $aa);
my ($pfile, $nfile) = @ARGV:
open(PROTEIN, "<$pfile") or die $!;
open(DNA, "<$nfile") or die $!;

while(my $line = <PROTEIN>) {
  chomp($line);
  if($line =~ m/^>/) {
    $name = shift split(' ', $line);
    next;
  }
  $proteins{$name}.= $line if $name;
}
close(PROTEIN);

for my $name (keys %proteins) {
  $proteins{$name} =~ s/\s+//sg;
}
while($line = <DNA>) {
  chomp($line);
  if($line =~ m/^>/) {
    $name =shift split(' ', $line);
    next;
  }
  $dnas{$name} .= $line if $name;
}
close(DNA);
```

```
for $name (keys %dna) {
  unless (exists $protein{$name}) {
    warn "DNA sequence $name has no
         matching protein sequence!\n";
    next;
  }
  $dna{$name} =~ s/\s+//sg;
  print "$name\n";
  my @protein=split('', $proteins{$name});
  my @dna = split('', $dnas{$name});
  for $aa (@protein) {
    if($aa eq '-') {
      print "---";
    } else {
      $codon = shift @dna
         . shift @dna . shift @dna;
      # check that $codon codes for $aa
      print $codon;
    }
  }
  print "\n";
}
```

Exercises

1. Write an extraction program to report
   (a)the most significant library hit
   (b)all significant library hits
   that result from a FASTA search. Several example search results are available at crick.med:/seqlib/bioch508/perl/fasta.result*
2. Write the same program for blast output. Sample results are in the same directory