#!/usr/bin/env node

let fs = require('fs');
let readline = require('readline');
let a = false;
let i = '';
let r = false;
function checkFile(file) {
    try {
        return fs.statSync(file).isFile();
    }
    catch (error)
    {
        return false;
    }
}
for(let k = process.argv.length-1; k >= 0; k--){
  if(process.argv[k] === '-a'){
    a = true;
  } else if(process.argv[k] === '-i'){
    if(checkFile(process.argv[k+1]) === true){
      i = process.argv[k+1];
    }
  } else if(process.argv[k] === '-r'){
    r = true;
  }
}
if(i.length >= 1){
    let o = '
A JavaScript for processing GenBank DNA FASTA files.

Usage:
' + '-a\t-reduces names to GenBank accessions only (default = ' + a + ')
' + '-i\tspecifies a GenBank input fasta file (required)
' + '-r\tproduces the reverse complement of each sequence (default = ' + r + ':'

')

    process.stderr.write(o, 'UTF8');
}

```javascript
function format(n, s) {
    if ((typeof n === 'string') && (typeof s === 'string')) {
        if (n.length > 2 && s.length > 1) {
            let labels = n.replace(/\>/, '').split(' ');
            let o = '>'
            if (a === true) {
                o += labels[0];
            } else {
                o += labels[1] + '_' + labels[2] + '_' + labels[0];
            }
            let b = s.toUpperCase().replace(/[^ACGTNVDBHWMRKSY]/g, '').split('');
            if (r === true) {
                b = b.join('').replace(/[^ACGTNVDBHWMRKSY]/g, function(x) {
                    return {
                        A: 'T',
                        C: 'G',
                        G: 'C',
                        T: 'A',
                        V: 'B',
                        D: 'H',
                        B: 'V',
                        H: 'D',
                        M: 'K',
                        R: 'Y',
                        K: 'M',
                        Y: 'R'
                    }[x]);
                }).split('').reverse();
            } else {
                l = b.length;
                for (let k = 0; k < l; k++) {
                    if ((k % 80) === 0) {
                        o += '\n' + b[k];
                    } else {
                        o += b[k];
                    }
                }
                return(o + '\n');
            }
        }
    }
}
```
return ('');
}

let line = readline.createInterface({
  input: fs.createReadStream(i),
  output: process.stdout,
  terminal: false
});
let n = '';
let nRE = new RegExp(/\^>/);
let s = '';
line.on('line', function(line){
  line = line.trim();
  if(nRE.test(line) === true){
    process.stdout.write(format(n, s), 'UTF8');
    s = '';
    n = line;
  } else {
    s += line;
  }
  return(false);
});
line.on('close', function(){
  process.stdout.write(format(n, s), 'UTF8');
  return(false);
});
DNA/RNA/protein sequences are ‘special’ text
  case is (usually) meaningless
  can be used to denote sequence/alignment quality
  orientation is not (usually) important
  sequences are archived in arbitrary orientation
  [protein sequences have just one orientation]
  commonly coded as letters, but could be numbers, etc.
‘query’ sequence used to find ‘reference’ sequence(s)
queries are entire sequence or sequence fragment(s)
...searching for sequences...

grep
  one reference sequence per line
  query twice (both orientations)
  will find exact matches only (if query is a substring)
  maybe not so useful

tre-agrep
  one reference sequence per line
  specify maximum allowable number of mismatches
  query twice (both orientations)
  similar results as megaBLAST (if query is a ‘substring’)
...searching for sequences...

hidden Markov model (e.g. HMMR)
- uses a probabilistic model of a sequence alignment
- or a model of one sequence using a substitution matrix
- finds similar sequences

machine learning (neural networks)
- a complex model of particular sequence types
- used to find features (e.g. splice sites)
...searching for sequences

a more robust sequence search
alignment of query (both orientations) to all references
  ‘global’ alignment (all positions aligned)
  ‘local’ alignment (core most similar positions aligned)
compute similarity
  edit distance (a.k.a. p–distance, raw distance)
similarity matrices
return the most similar sequences/fragments
effective, but slow (SEQHP: Goad and Kanehisa 1982)
Figure 2. A forward path matrix for the comparison of bases 1050 to 1150 and 1680 to 1780 within the 5495 base segment of the mouse immunoglobulin κ light chain gene complex (Ref. 9).
Figure 3. The corresponding reverse path matrix.
Figure 4. The logical product of the two path matrices.
FASTP and FASTA...

Lipman and Pearson (1985); Pearson and Lipman (1988)

the ‘ancestor’ of BLAST

search for likely ‘similar’ sequences

use a pre–computed table of oligo by sequence position

compute offsets to find clustered similarities (diagonals)

estimate sequence similarity (number of shared oligos)

pick best segments

join segments, recalculate similarity with substitutions

local alignment including best segments
<table>
<thead>
<tr>
<th>seq</th>
<th>oligo0</th>
<th>oligo1</th>
<th>oligo2</th>
<th>oligo3</th>
<th>oligo4</th>
<th>oligo5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>1,28</td>
<td>20,22</td>
<td>15</td>
<td>5</td>
<td>2,10</td>
<td>12</td>
</tr>
<tr>
<td>0</td>
<td>5,32</td>
<td>24,26</td>
<td>19</td>
<td>9</td>
<td>6,14</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>–</td>
<td>28,13</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>7,11</td>
</tr>
<tr>
<td>3</td>
<td>4,31</td>
<td>23,25</td>
<td>18</td>
<td>–</td>
<td>5,13</td>
<td>15,32</td>
</tr>
</tbody>
</table>
# oligo offsets

<table>
<thead>
<tr>
<th>seq</th>
<th>oligo0</th>
<th>oligo1</th>
<th>oligo2</th>
<th>oligo3</th>
<th>oligo4</th>
<th>oligo5</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>0,0</td>
<td>0,0</td>
<td>0</td>
<td>0</td>
<td>0,0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>4,4</td>
<td>4,4</td>
<td>4</td>
<td>4</td>
<td>4,4</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>0,–</td>
<td>–</td>
<td>13,–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>-19</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>-5,–</td>
</tr>
<tr>
<td>3</td>
<td>3,3</td>
<td>3,3</td>
<td>3</td>
<td>–</td>
<td>3,3</td>
<td>3,–</td>
<td>...</td>
</tr>
</tbody>
</table>
Fig. 1. Identification of sequence similarities by FASTA. The four steps used by the FASTA program to calculate the initial and optimal similarity scores between two sequences are shown. (A) Identify regions of identity. (B) Scan the regions using a scoring matrix and save the best initial regions. Initial regions with scores less than the joining threshold (27) are dashed. The asterisk denotes the highest scoring region reported by FASTP. (C) Optimally join initial regions with scores greater than a threshold. The solid lines denote regions that are joined to make up the optimized initial score. (D) Recalculate an optimized alignment centered around the highest scoring initial region. The dotted lines denote the bounds of the optimized alignment. The result of this alignment is reported as the optimized score.

(Pearson and Lipman 1988)
aa substitution matrices...

derived from ‘curated’ multiple sequence alignments
highly sample dependent
assume alignment is correct, sequences are homologous
PAM (Dayhoff, Schwartz, and Orcutt 1978)
  Point Accepted Mutation
PAMx: $x = x$ mutations per 100 amino acids
derived from alignments of protein ‘families’
BLOSUM (Henikoff and Henikoff 1992)

BLOcks of Amino Acid SUbstitution Matrix

BLOSUMx: $x =$ similarity used to merge sequences prior to making the matrix

higher numbers are for use with more similar sequences
derived from indel free alignment segments

original BLOSUM62 has an error, use corrected version
nt substitution matrices

formulae can be used, but generally are not
JC69 (Jukes and Cantor 1969), K80 (K2P; Kimura 1980),
HKY85 (Hasegawa, Kishino, and Yano 1985), T92
(Tamura 1992), GTR (Tavaré 1986)
nucleotide matrices not usually based on empirical data
equal weights commonly used
+5/-4 or +1/-2 (match vs. mismatch)

```
ATGCCTGCACGC
| | | | | | | | |
ATGCATGCATGC
555545555555
= 50-8
= 42 (70%)
```

```
ATGCCTGCACGC
| | | | | | | | |
ATGCATGCATGC
1111211111211
= 10-4
= 6 (50%)
```
...FASTP and FASTA...

assessment of statistical significance
using RDF or RDF2
permute (shuffle) reference sequences
  the entire database or just similar sequences (faster)
  number of positions and base composition constant
  order jumbled
recalculate similarity
p–values are not calculated (distribution not normal)
z–values used instead

\[ z = \frac{\text{similarity} - \text{mean of permuted similarity}}{\text{standard deviation of permuted similarity}} \]

- \( z > 3 \) = possibly significant (0.0013%)
- \( z > 6 \) = probably significant
- \( z > 10 \) = significant
Table 3. Statistical significance (z value) of protein similarity scores. Protein sequences from the searches discussed in examples 1, 2, and 3 were compared with the best related and unrelated library sequences found. Z values [(score – mean score)/standard deviation] were calculated for the initial score from the mean and standard deviation of the database initial scores (initial scan), and for the initial (I) and optimized (O) scores from the mean and standard deviation of scores against randomly permuted versions of the database sequence in question. In the latter case, 50 comparisons (ktup = 1) were made with shuffled sequences.

<table>
<thead>
<tr>
<th>Query sequence</th>
<th>Library sequence matched</th>
<th>Initial scan</th>
<th>Randomized I</th>
<th>Randomized O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifier</td>
<td>Protein</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OKBO2C (bovine cyclic AMP kinase)</td>
<td>TVBY8</td>
<td>Yeast cell cycle control</td>
<td>11.3</td>
<td>11.1</td>
</tr>
<tr>
<td></td>
<td>TGVFV-R</td>
<td>Src</td>
<td>11.0</td>
<td>10.4</td>
</tr>
<tr>
<td></td>
<td>TVMS M</td>
<td>Mos</td>
<td>9.3</td>
<td>7.6</td>
</tr>
<tr>
<td>ANRT (rat angiotensinogen)</td>
<td>ITHU</td>
<td>Alpha-1 antitrypsin</td>
<td>10.0</td>
<td>8.3</td>
</tr>
<tr>
<td></td>
<td>G1HUNM</td>
<td>Human Ig heavy chain (V-2)</td>
<td>6.8</td>
<td>8.1</td>
</tr>
<tr>
<td></td>
<td>XHHU3</td>
<td>Antithrombin</td>
<td>5.3</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>ITHUC</td>
<td>Alpha-1 antichymotrypsin</td>
<td>5.0</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>TVMV-S</td>
<td>PDGF-related sis</td>
<td>4.6</td>
<td>3.8</td>
</tr>
<tr>
<td>VHVUNH (snowshoe hare bunyavirus nucleoprotein)</td>
<td>ORBPL</td>
<td>Lambda replication protein</td>
<td>4.4</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>GHRB</td>
<td>Rabbit Ig gamma C region</td>
<td>4.3</td>
<td>4.8</td>
</tr>
</tbody>
</table>

(Lipman and Pearson 1985)
FASTA vs. BLAST

BLAST has larger default word size (== faster)
  can be less ‘sensitive’ (ignores low similarly sequences)
BLAST focuses on most informative $k$–mers
improved treatment of indels
  ‘small’ indels and/or ‘short’ segments of mismatch between matching words are tolerated
different methods of calculating significance
  BLAST provides a ‘significance’ of match assuming that search settings are correct
BLAST flavors...

Altschul et al. (1990, 1997)
the ‘original’ source of algorithms
1997 overcomes non–matching regions (e.g. indels)
freely available from NCBI (binary and code)

Kent (2002; BLAT)
faster in some circumstances (e.g. large batches)
designed to search against whole genomes
freely available from the author (binary and code)

Gish (2006; WU-BLAST)
same algorithm as NCBI BLAST, but faster
different default settings => different results
binary (possibly available), but code is not distributed
Nguyen and Lavenier (2009; PLAST) parallel version
3–6× faster than multithreaded BLAST
freely available from the authors (binary and code)

Panagiotis and Sahinidis (2011; GPU-BLAST)
requires a CUDA graphics processor
up to 10× faster
protein sequences only
freely available from the authors (binary and code)
Zhao and Chu (2014; G-BLASTN) requires a CUDA graphics processor up to 14× faster. Nucleotide sequences only are inefficient on short sequences (> 1,000,000 bp). The database must fit into the GPU memory and is freely available from the authors (binary and code).