Homework 7

```
yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | head -5
9606    1       2591067
9606    1       3458201
9606    1       3610142
9606    1       8889549
9606    1       12477932
```

How many homo sapiens (using human taxid) genes are there?

```
yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | cut -f2 | sort -u
| wc -l
31386
```

The top 10 genes with the largest number of pubmed paper

```
yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | cut -f2 | sort |
uniq -c | sort –k 1,1nr | head -5
  5662 7157
  3900 7124
  3074 348
  2736 1956
  2723 7316
```

A good way to understand a long command line:

Run each step and less to see what happened and then add the next step and less

$$\underbrace{\texttt{less gene2pubmed}}_{1}\ \texttt{|}\ \underbrace{\texttt{awk '\$1==9606'}}_{2}\ \texttt{|}\ \underbrace{\texttt{cut -f2}}_{3}\ \texttt{|}\ \underbrace{\texttt{sort}}_{4}\ \texttt{|}\ \underbrace{\texttt{uniq -c}}_{5}\ \texttt{|}\ \underbrace{\texttt{sort -k 1,1nr}}_{6}\ \texttt{| less}$$

```
yyin@glu:~/work/class$ less gene2pubmed

yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | less

yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | cut -f2 | less

yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | cut -f2 | sort | less

yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | cut -f2 | sort | uniq
-c | less

yyin@glu:~/work/class$ less gene2pubmed | awk '$1==9606' | cut -f2 | sort | uniq
-c | sort -k 1,1nr | head -5
```

Homework #10

1. In homework #9, you got fasta sequences of GenBank GH5 proteins.
- Design command line to extract sequence IDs and save as separate file.
- Using this ID file as input, write a simple perl script to generate a color definition file with 1st col to be the protein ID and the 2nd col to be the hex blue color code (ref to slide 14 of this class).

2. In homework #9, you also got the output files of searching GH5 homologous proteins against cow rumen metagenome using three search tools (blast, fasta and hmmer).
- Modify the command line in slide #9 of http://cys.bios.niu.edu/yyin/teach/PBB/Apr02-2013.pdf to extract the hit IDs (use the shown e-value and coverage cutoffs in the slide) and save as separate file.
- Use the example perl script that I have in this class (slide 24) to extract the fasta sequences from cow rumen metagenome database (two input files: ID file and fasta seq file).

3. Similar to 1, prepare a color definition file for metagenome hit IDs using hex red color. Combine the two definition files into one using cat.

4. Combine the fasta sequences from GenBank (step 1) and metagenome (step 2).

5. For combined fasta sequences from 4, MAFFT alignment and then FASTTREE to build phylogeny.

6. Upload newick tree file in 5 and the combined color definition file in 3 to iTOL to generate colored phylogeny and save the radiation view as a PDF figure.

Report due April 23 (send by email)

Office hour:
Tue, Thu and Fri 2-4pm, MO325A
Or email: yyin@niu.edu

Create a file called math.pl
perl math.pl

If use vi
Press i to the edit mode
Esc then :x to save and exit
Esc then :q! to exit without save

```
#!/usr/bin/perl -w

$x=3;  # assign value to scalar
$y=2;

print "$x plus $y is equal to ", $x+$y ,"\n";
$sum=$x+$y;
print "\$sum is equal to $sum \n";

print "$x minus $y is equal to ", $x-$y ,"\n";
$minus=$x-$y;
print "\$minus is equal to $minus \n";

print "$x times $y is equal to ", $x*$y ,"\n";

print "$x divided by $y is equal to ", $x/$y ,"\n";
```

Array and hash are VERY useful to hold text data in the memory and are often created using loops

- Creating Array:

```
@fruit_list = ('apple', 'orange', 'banana');
```

=

```
$fruit_list[0]='apple';
$fruit_list[1]='orange';
$fruit_list[2]='banana';
```

- Creating Hash:

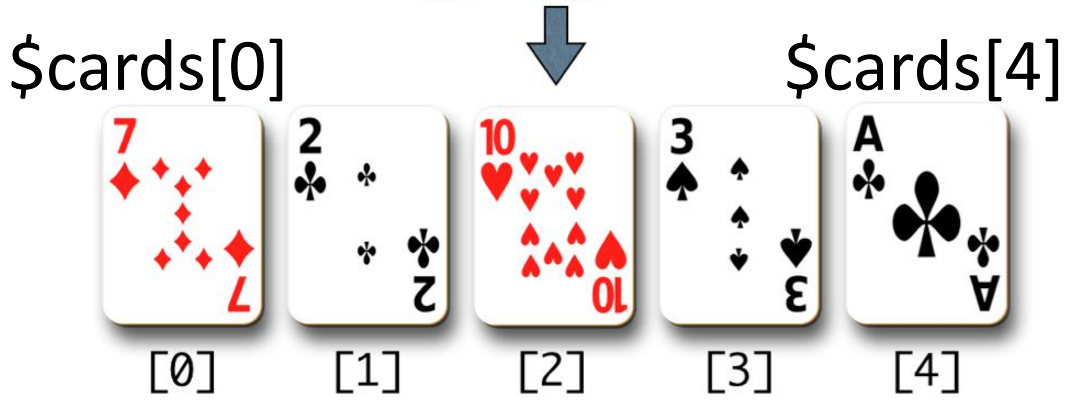
```
%ip2hostname = (
"glu" => "131.156.41.196",
"gly" => "131.156.41.193",
"cys" => "131.156.41.195"
);
```

=

```
$ip2hostname{"glu"}='131.156.41.196';
$ip2hostname{"gly"}='131.156.41.193';
$ip2hostname{"cys"}='131.156.41.195';
```

Array is indexed by numbers beginning from 0

`@cards = (7, 2, 10, 3, ‘A’);`



Function	Meaning
push(@array, "some value")	add a value to the end of the list
$popped_value = pop(@array)	remove a value from the end of the list
$shifted_value = shift (@array)	remove a value from the front of the list
unshift(@array, "some value")	add a value to the front of the list
splice(...)	everything above and more!

http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v3.1.1.html

% ip2hostname

glu	131.156.41.196
gly	131.156.41.193
cys	131.156.41.195

```
@server = keys %ip2hostname
```

@server will have three server names (‘glu’, ‘gly’, ‘cys’)

Function	Meaning
keys %hash	returns an array of keys
values %hash	returns an array of values
exists $hash{key}	returns true if the key exists
delete $hash{key}	removes the key and value from the hash

Create an array from a tabular format file using perl

```
vi array-from-file.pl
```

```
#!/usr/bin/perl -w

open (IN,$ARGV[0]);

@a=<IN>;   # assign to an array

foreach $line (@a){
   print $line;
}
```

Save and exit vi

```
less /home/yyin/work/class/cosmicRaw.txt.head10.6col

perl  array-from-file.pl /home/yyin/work/class/cosmicRaw.txt.head10.6col
```

What happened?
File name is captured by an internal special variable and passed to a file handle

Then file content is written to an array and stored in the memory

File handle: IN, a temporary name assigned to a file.

@ARGV: special variable to capture command line arguments

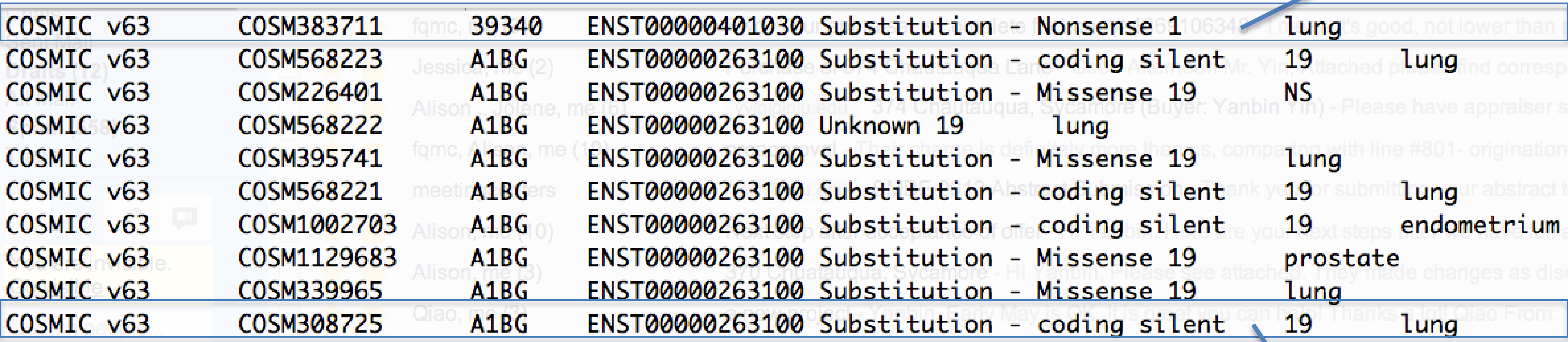
$ARGV[0]: the first element

```perl
#!/usr/bin/perl -w

open (IN,$ARGV[0]);

@a=<IN>;  # assign to an array

print $a[0];
print $a[-1];
```

Including the new line character

The entire file content is now in @a

$a[0]



```
COSMIC v63    COSM383711    39340   ENST00000401030 Substitution - Nonsense 1     lung
COSMIC v63    COSM568223    A1BG    ENST00000263100 Substitution - coding silent   19     lung
COSMIC v63    COSM226401    A1BG    ENST00000263100 Substitution - Missense 19     NS
COSMIC v63    COSM568222    A1BG    ENST00000263100 Unknown 19       lung
COSMIC v63    COSM395741    A1BG    ENST00000263100 Substitution - Missense 19     lung
COSMIC v63    COSM568221    A1BG    ENST00000263100 Substitution - coding silent   19     lung
COSMIC v63    COSM1002703   A1BG    ENST00000263100 Substitution - coding silent   19     endometrium
COSMIC v63    COSM1129683   A1BG    ENST00000263100 Substitution - Missense 19     prostate
COSMIC v63    COSM339965    A1BG    ENST00000263100 Substitution - Missense 19     lung
COSMIC v63    COSM308725    A1BG    ENST00000263100 Substitution - coding silent   19     lung
```

$a[9] or $a[-1]

```
less /home/yyin/work/class/cosmicRaw.txt.head10.6col
```

Something need to be repeated done for each element in the array

```
foreach $line (@a){
   print $line;
}
```

```
foreach $line (@a){
   print $line;
}
```

=

```
foreach(@a){
   print $_;
}
```

```
foreach(@a){
   @col=split(/\t/,$_);
   print $col[1],"\n";
}
```

Loop: read one element from an array at a time and process it

The element is renamed as $line and passed into the loop

Inside the loop, print the element, which is each of the rows in the file

Instead of renaming each element using a new variable, use the internal special variable $_ to capture each element. This makes the code easy to write but not easy to read

Create a new array @col:
Split is a new function, here to split a string into pieces given a field delimiter. "\t" is tabular space.

There are a lot of built-in perl functions that are useful for text processing, just like functions for numeric calculations (sqrt, log, abs, sin, cos etc.)

Perl has many predefined special variables that contain default values designed to **make life easier for programmers**. Most special variables are a combination of punctuation marks and obscure characters, and a programmer following the good coding practice of creating meaning variable names will never accidentally run into them.

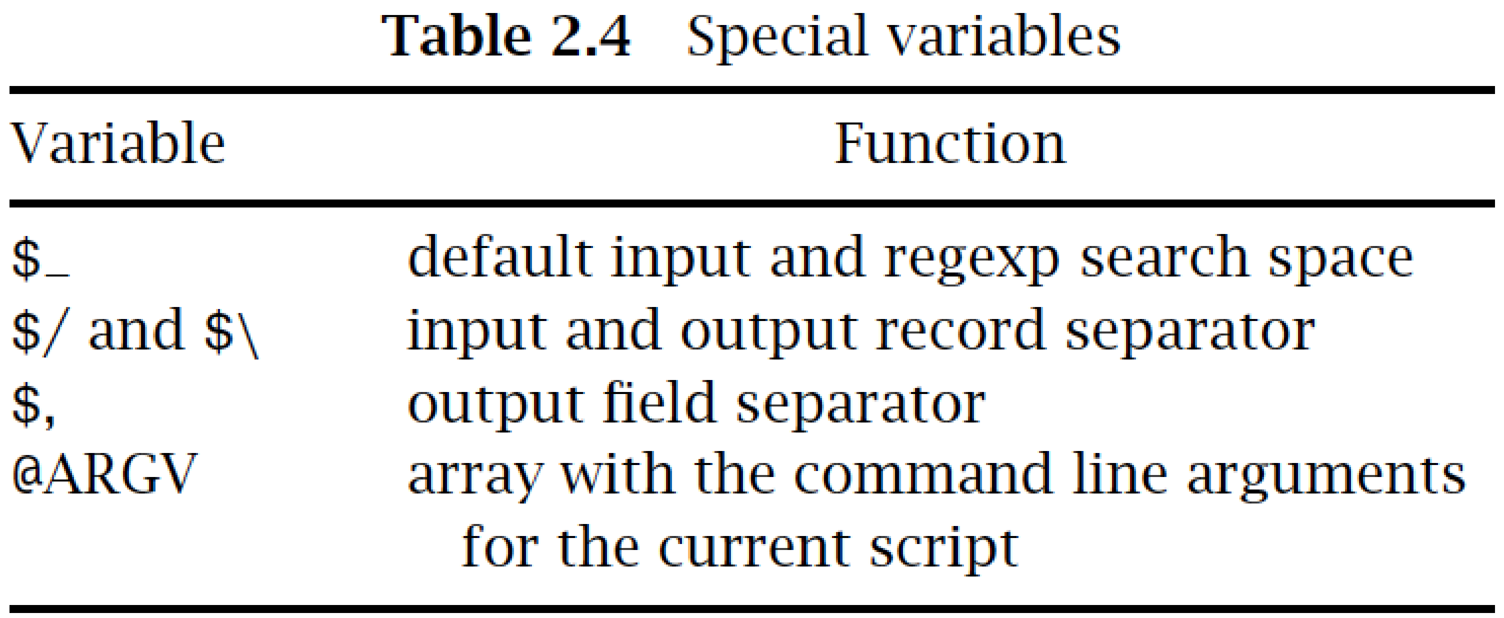


Table 2.4 Special variables

Variable	Function
$_	default input and regexp search space
$/ and $\	input and output record separator
$,	output field separator
@ARGV	array with the command line arguments for the current script

```
#!/usr/bin/perl -w

open (IN,$ARGV[0]);

@a=<IN>;   # assign to an array

foreach $line (@a){
   print $line;
}
```

Easier way to read in file without file handle

The "diamond operator", <> is used when a program is expecting input;

<> means perl accept data from standard input <STDIN> in the command line

||

```
#!/usr/bin/perl -w

while($line=<>){
  print $line;
}
```

=

```
#!/usr/bin/perl -w

while(<>){
  print $_;
}
```

while loop check a condition first (here if there are data coming in from a file handle, again one line at a time) and then get into the loop

```
perl  array-from-file2.pl /home/yyin/work/class/cosmicRaw.txt.head10.6col

cat /home/yyin/work/class/cosmicRaw.txt.head10.6col | perl  array-from-file2.pl
```

```
#!/usr/bin/perl

open (IN,$ARGV[0]);

@a=<IN>;

foreach(@a){
   @col=split(/\t/,$_);
   print $col[1],"\tmutation\n";
}
```

||

```
#!/usr/bin/perl

while (<>){
   @col=split(/\t/,$_);
   print $col[1],"\tmutation\n";
}
```

There's More Than One Way To Do It

<u>The pro of while:</u>
No need to load all data into memory; process data on a line by line basis

<u>The con of while:</u>
Can not reference other lines and can only work once

=

```
cat cosmicRaw.txt.head10.6col |
cut -f2 | awk '{print
$1,"mutation"}' | sed 's/ /\t/'
```

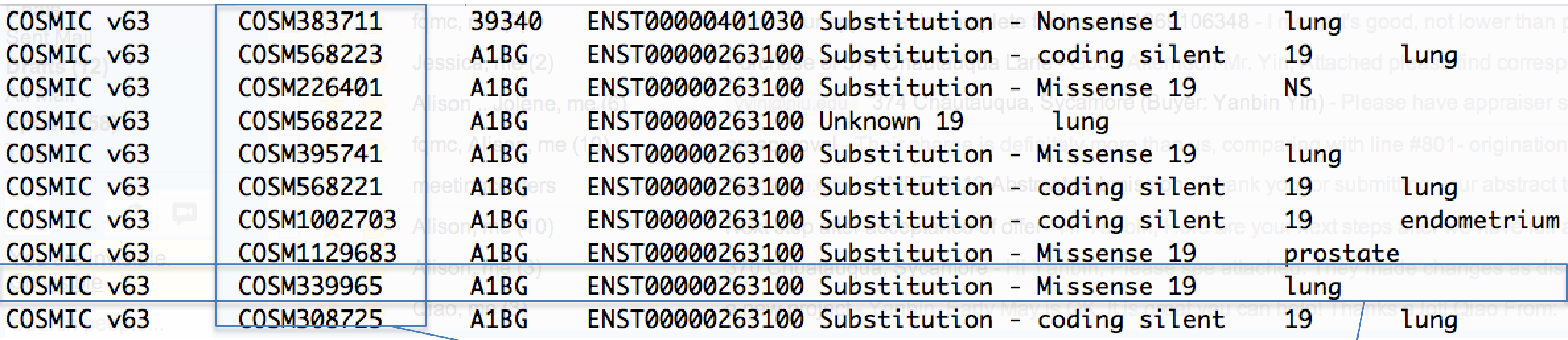
```
perl  array-from-file3.pl /home/yyin/work/class/cosmicRaw.txt.head10.6col
```

hash: name-value pair, name also called key

$hash{‘key’}=‘value’;

Like array, key and value of a hash can be assigned in a loop

Key and value could have strict one-to-one correspondence or not



```
COSMIC v63    COSM383711   39340  ENST00000401030 Substitution - Nonsense 1      lung
COSMIC v63    COSM568223   A1BG   ENST00000263100 Substitution - coding silent   19      lung
COSMIC v63    COSM226401   A1BG   ENST00000263100 Substitution - Missense 19     NS
COSMIC v63    COSM568222   A1BG   ENST00000263100 Unknown 19     lung
COSMIC v63    COSM395741   A1BG   ENST00000263100 Substitution - Missense 19     lung
COSMIC v63    COSM568221   A1BG   ENST00000263100 Substitution - coding silent   19      lung
COSMIC v63    COSM1002703  A1BG   ENST00000263100 Substitution - coding silent   19      endometrium
COSMIC v63    COSM1129683  A1BG   ENST00000263100 Substitution - Missense 19     prostate
COSMIC v63    COSM339965   A1BG   ENST00000263100 Substitution - Missense 19     lung
COSMIC v63    COSM308725   A1BG   ENST00000263100 Substitution - coding silent   19      lung
```

```
#!/usr/bin/perl -w

while(<>){
  @col=split(/\t/,$_);
  $cosmic{$col[1]}=$_;
}
print $cosmic{'COSM339965'};
```

Key: $col[1], the second col
Value: $_, the entire row

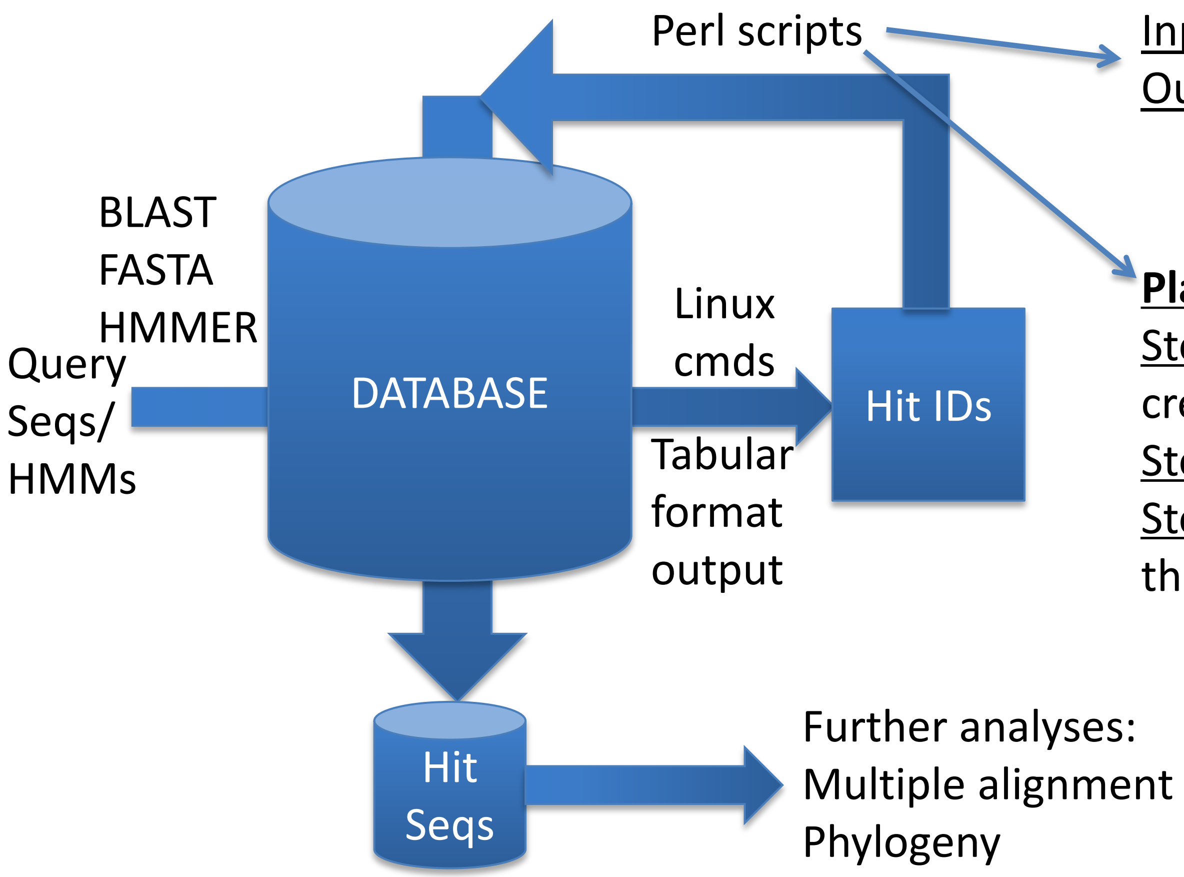
Call this particular key

```
perl hash-from-file.pl cosmicRaw.txt.head10.6col
```

This could be VERY useful for holding fasta sequences: id as the key and sequence as the value

We can store a large fasta file in the memory and call any specific ids anytime

e.g. read in a list of subset IDs to extract the sequences



Input: id file and fasta seq file
Output: a smaller fasta seq file

Planning:
Step 1: read in fasta seq file to create hash (load into memory)
Step 2: read in the subset id file
Step 3: use the id as the key to call the value (sequence)

do the following in /media/DATAPART1/z1576493/class/mar19/

```
formatdb -i ecoli-all.faa
formatdb - # see the options, for nt db, also use -p F
less ecoli-all.faa # select the 3rd protein sequence(YP_488309.1)
vi test-query.fa # create a file to store this protein seq

[now blast, which is in your path already]
blastall -p blastp -i test-query.fa -d ecoli-all.faa
blastall -p blastp -i test-query.fa -d ecoli-all.faa > test-query.fa.out

[-m 9, the tabular format output without alignment, easy to parse]
blastall -p blastp -i test-query.fa -d ecoli-all.faa -m 9
blastall -p blastp -i test-query.fa -d ecoli-all.faa -m 9 > test-
query.fa.out.m9

[-e 1e-2, showing only hits with evalue < 1e-2]
blastall -p blastp -i test-query.fa -d ecoli-all.faa -m 9 -e 1e-2

[Now try something big (and slow)]
time blastall -p blastp -i test-query.fa -d
/home/yyin/work/class/metagenemark_predictions.faa -m 9 -e 1e-2 > test-
qery.fa.cowrumen.out.m9 &

[Do some parsing]
less test-query.fa.cowrument.out.m9 | cut -f1,2,3,7- | less
less test-query.fa.cowrument.out.m9 | cut -f1,2,3,7- | grep -v '^#' |
cut -f2 | sort -u | head
```

```
# Save a test id file
less test-query.fa.cowrument.out.m9 | cut -f1,2,3,7- | grep -v '^#' | cut -
f2 | sort -u | head > test-query.fa.cowrument.out.m9.head10.id

# Check a smaller fasta seq file
less test-query.fa.cowrument.out.m9.head10.fa
```

```
>NODE_27_length_1627_cov_2.424708_orf_00100 2..1597
SLPSMRADSFTKELMEKISSVRTSTLTFAPEAGTPRLRDIINKNITEEEILRACR
VAYEAGKNQIKLYFMDGLPGETYEDIAGIAALASHVVDEYYRTPGRNKARQP
QVTLSVACFIPKPHTPFQWERQNAPEELADKQAFLSGKITDRKVRHNYHDA
KVSRIEAVFARGDRRLGRALEEAARRHVRFDAWEDCFDYDGWMDIFETVG
IDPAFYANRTIPDDEILPWDMISCGVTKSFLLSERHKAQQAIATPACRDQCSG
CGVNRLVDKRYCRWCPGHPESSDSAGRITSDREIRKKPEETSAQKGNVKPAR
QIRIRFRKYGAMLYISHLDLAKTVMRSIVRSGLPVYYSEGFNPKPKLVFGTPLS
VGCGGEAEVLDIRLMKAVSNAEITEKLKAVMPNGVEVTQVYEQKGKLTDVK
WAENVIEWRNTDVSPELAEKTEALFQSPVVMMKKSKSGEKEVDITSYIRSLR
AEALDGGLRITAVTAAEQENYLNPEYIVQAAERAFGISGENGWHVITRTRLLL
ADGETDFA*
>NODE_33_length_1571_cov_1.473584_orf_00110 1..186
GVVTAKDADVTSAPNNKSQTLNTLSEGTTFEVLSEQGGFVEIRLGEKIRGFVK
TSDVGIVK*
>NODE_33_length_1571_cov_1.473584_orf_00120
complement(218..991)
MVKRGENQLSLMQKFLCALLLALCCNAFATESSGDDSSSYDDQAWRNSKK
YKTWKKYSERDVHAPKALEFRVAGMYPTAFDASVLAFRAVNLVEINDRWRF
YVGYDPFHVTYNEKGFSDESLMLVGAVLAISPFTLIYSAIKGSGSRDPAEEMN
DYYKEASIPKIIFFYIPAYIWCGNLYFPLVEGSWLGLNDQSHVVTHIIEEGGFYL
RSFTYTNDVSLRFSKSGYFVDAGVRLEKNFADDFKARIILQIGVFGSG*
```

Description line does not allow to break into multiple lines (have multi- newline character)

Sometimes there are multiple lines (newline)

Step 1: process fasta seq file to create hash (load into memory)

We want to get ride of newline character inside the sequences

```
vi get-seq.pl
```

Here is where the second entry started

The tabular space

```
#!/usr/bin/perl -w

while(<>){
   chomp $_;  # get ride of newline character
   if($_=~/>/){  # =~ is used to match regexp
     $_=~s/>//;  # =~s is used to substitute
     print $_,"\t"; # insert a tabular space
   }
   else{
     print $_; # print the sequence line (no
               # newline character anymore)
   }
}
```

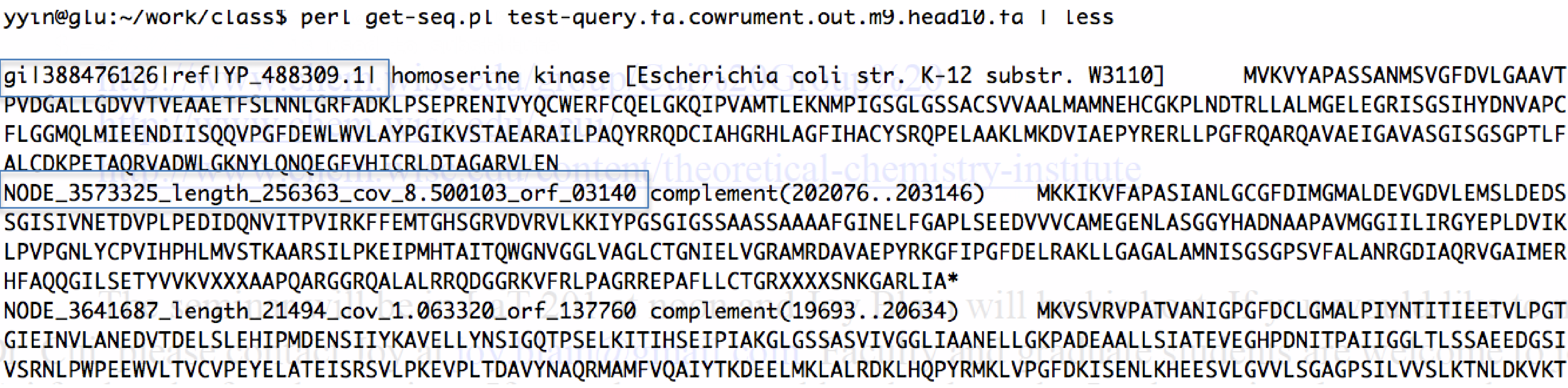
if/else statement to control the flow

```
if (condition)
{
  action;
}
elsif
(condition){
  action;
}
else{
  action;
}
```

```
perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa
```

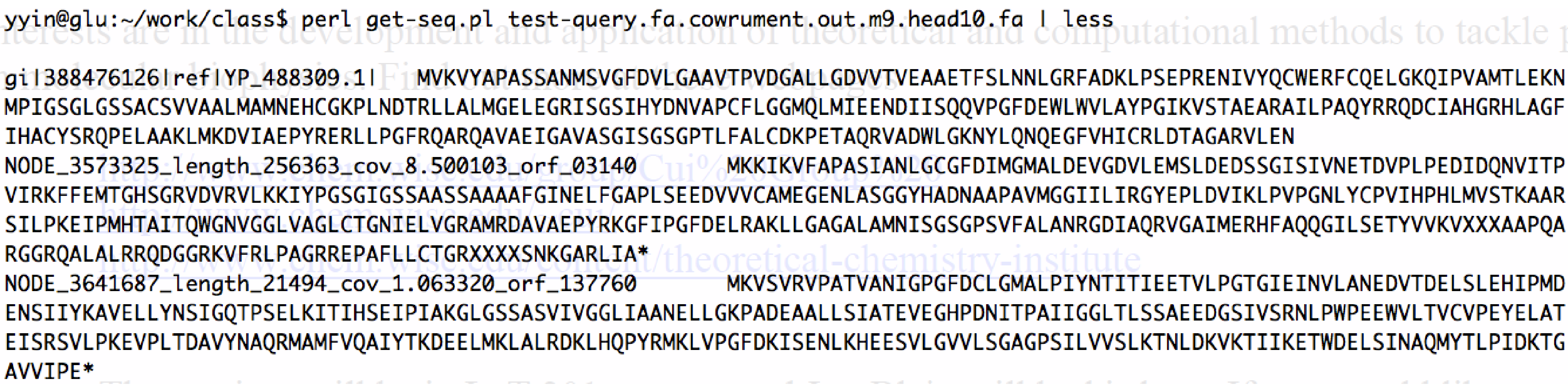
```
gi|388476126|ref|YP_488309.1| homoserine kinase [Escherichia coli str. K-12 substr. W3110]    MVKVYAPASSANMSVGFDVLGAAVT
PVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPC
FLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGFIHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLF
ALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLENNODE_3573325_length_256363_cov_8.500103_orf_03140 complement(202076..203146)
      MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDSSGISIVNETDVPLPEDIDQNVITPVIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELFGAP
LSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIKLPVPGNLYCPVIHPHLMVSTKAARSILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGF
IPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIMERHFAQQGILSETYVVKVXXXAAPQARGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
NODE_3641687_length_21494_cov_1.063320_orf_137760 complement(19693..20634) MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGTGIEIN
VLANEDVTDELSLEHIPMDENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGGLIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSIVSRNL
PWPEEWVLTVCVPEYELATEISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELMKLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDKVKTIIKET
WDELSINAQMYTLPIDKTGAVVIPE*NODE_3701631_length_68488_cov_2.143266_orf_51600 complement(15723..16385)    MGVSAFAPASIGNVSVG
FDILGAALKPIDGQILGDNVDVYAGDSEFDLSIEGWFASKLPADPKKNICYDAYVGFKALLEEKGIAVKPVKMVLKKNLPIGSGLGSSAASIVAAVEALNAFHDYPLSKDEALTLMGRLEG
```

```perl
#!/usr/bin/perl -w

while(<>){
   chomp $_;  # get ride of newline character
   if($_=~/>/){  # =~ is used to match regexp
     $_=~s/>//;  # =~s is used to substitute
     print "\n",$_,"\t"; # insert a tabular space
   }
   else{
     print $_; # print the sequence line (no
                    # newline character anymore)
   }
}
```



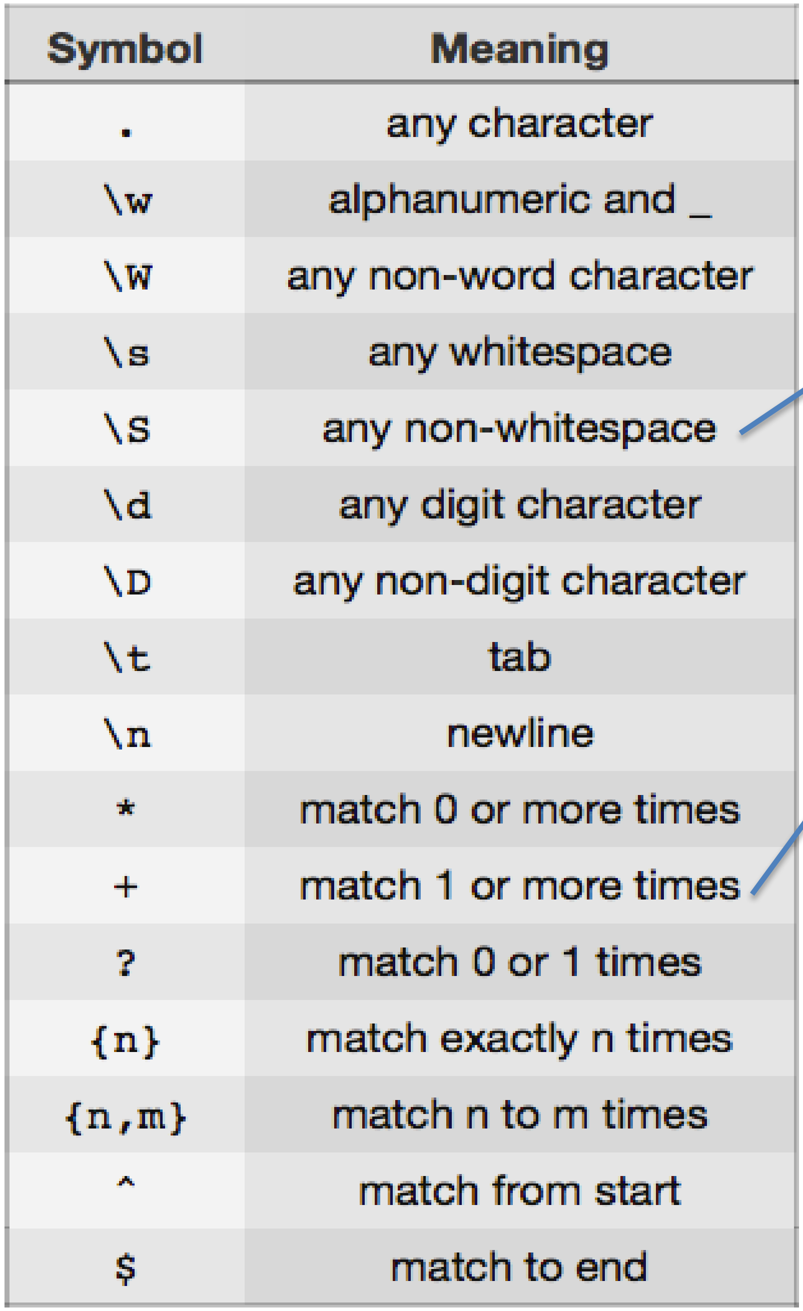
```
yyin@glu:~/work/class$ perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa | less

gi|388476126|ref|YP_488309.1| homoserine kinase [Escherichia coli str. K-12 substr. W3110]      MVKVYAPASSANMSVGFDVLGAAVT
PVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPC
FLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGFIHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLF
ALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLEN
NODE_3573325_length_256363_cov_8.500103_orf_03140 complement(202076..203146)    MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDS
SGISIVNETDVPLPEDIDQNVITPVIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELFGAPLSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIK
LPVPGNLYCPVIHPHLMVSTKAARSILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGFIPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIMER
HFAQQGILSETYVVKVXXXAAPQARGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
NODE_3641687_length_21494_cov_1.063320_orf_137760 complement(19693..20634)    MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGT
GIEINVLANEDVTDELSLEHIPMDENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGGLIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSI
VSRNLPWPEEWVLTVCVPEYELATEISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELMKLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDKVKT
```

```
#!/usr/bin/perl -w

while(<>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/); # regexp, match a portion of text
                          # and capture the matched by ()
    print “\n”,$id,"\t"; # insert a tabular space
  }
  else{
    print $_; # print the sequence line (no
        # newline character anymore)
  }
}
```



```
yyin@glu:~/work/class$ perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa | less

gi|388476126|ref|YP_488309.1|   MVKVYAPASSANMSVGFDVLGAAVTPVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKN
MPIGSGLGSSACSVVAALMAMNEHCGKPLNDTRLLALMGELEGRISGSIHYDNVAPCFLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGF
IHACYSRQPELAAKLMKDVIAEPYRERLLPGFRQARQAVAEIGAVASGISGSGPTLFALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLEN
NODE_3573325_length_256363_cov_8.500103_orf_03140       MKKIKVFAPASIANLGCGFDIMGMALDEVGDVLEMSLDEDSSGISIVNETDVPLPEDIDQNVITP
VIRKFFEMTGHSGRVDVRVLKKIYPGSGIGSSAASSAAAAFGINELFGAPLSEEDVVVCAMEGENLASGGYHADNAAPAVMGGIILIRGYEPLDVIKLPVPGNLYCPVIHPHLMVSTKAAR
SILPKEIPMHTAITQWGNVGGLVAGLCTGNIELVGRAMRDAVAEPYRKGFIPGFDELRAKLLGAGALAMNISGSGPSVFALANRGDIAQRVGAIMERHFAQQGILSETYVVKVXXXAAPQA
RGGRQALALRRQDGGRKVFRLPAGRREPAFLLCTGRXXXXSNKGARLIA*
NODE_3641687_length_21494_cov_1.063320_orf_137760       MKVSVRVPATVANIGPGFDCLGMALPIYNTITIEETVLPGTGIEINVLANEDVTDELSLEHIPMD
ENSIIYKAVELLYNSIGQTPSELKITIHSEIPIAKGLGSSASVIVGGLIAANELLGKPADEAALLSIATEVEGHPDNITPAIIGGLTLSSAEEDGSIVSRNLPWPEEWVLTVCVPEYELAT
EISRSVLPKEVPLTDAVYNAQRMAMFVQAIYTKDEELMKLALRDKLHQPYRMKLVPGFDKISENLKHEESVLGVVLSGAGPSILVVSLKTNLDKVKTIIKETWDELSINAQMYTLPIDKTG
AVVIPE*
```

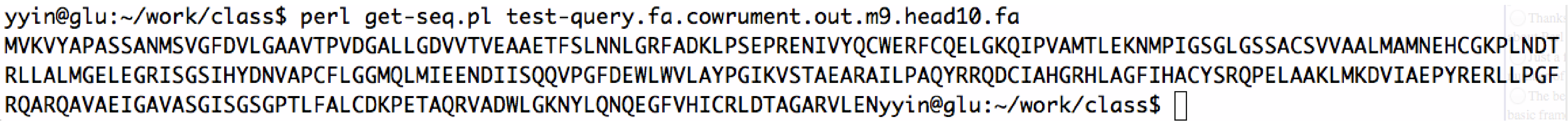
regexp metacharacters



Symbol	Meaning
.	any character
\w	alphanumeric and _
\W	any non-word character
\s	any whitespace
\S	any non-whitespace
\d	any digit character
\D	any non-digit character
\t	tab
\n	newline
*	match 0 or more times
+	match 1 or more times
?	match 0 or 1 times
{n}	match exactly n times
{n,m}	match n to m times
^	match from start
$	match to end

```
($id)=($_=~/(^\S+)/);
```

```
#!/usr/bin/perl -w

open(IN,$ARGV[0]);
while(<IN>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){  # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/);
#    print "\n",$id,"\t"; # insert a tabular space
    $seq_hash{$id}=$seq; # creating hash id-seq pairs
    $seq="";  # empty the $seq variable
  }
  else{
     $seq=$seq.$_; # concatenate seq fragments that was
                   # separated by newlines
#    print $_; # print the sequence line (no
# newline character anymore)
  }
}

print $seq_hash{'NODE_3573325_length_256363_cov_8.500103_orf_03140'};
```



```
yyin@glu:~/work/class$ perl get-seq.pl test-query.fa.cowrument.out.m9.head10.fa
MVKVYAPASSANMSVGFDVLGAAVTPVDGALLGDVVTVEAAETFSLNNLGRFADKLPSEPRENIVYQCWERFCQELGKQIPVAMTLEKNMPIGSGLGSSACSVVAALMAMNEHCGKPLNDT
RLLALMGELEGRISGSIHYDNVAPCFLGGMQLMIEENDIISQQVPGFDEWLWVLAYPGIKVSTAEARAILPAQYRRQDCIAHGRHLAGFIHACYSRQPELAAKLMKDVIAEPYRERLLPGF
RQARQAVAEIGAVASGISGSGPTLFALCDKPETAQRVADWLGKNYLQNQEGFVHICRLDTAGARVLENyyin@glu:~/work/class$ 
```

```
#!/usr/bin/perl -w

open(DB,$ARGV[0]);
open(OUT,">tmp");

while(<DB>){
  chomp $_;  # get ride of newline character
  if($_=~/>/){   # =~ is used to match regexp
    $_=~s/>//;  # =~s is used to substitute
    ($id)=($_=~/(^\S+)/);
    print OUT "\n",$id,"\t"; # insert a tabular space
  }
  else{
    print OUT $_; # print the sequence line
  }
}
close OUT; close DB;

open(IN,"tmp");
while (<IN>){
  next if $_=~/^$/;
  chomp $_;
  @col=split(/\t/,$_);
  $seq_hash{$col[0]}=$col[1];
}
close IN;
open(ID,$ARGV[1]);
while(<ID>){
  chomp $_;
  print ">$_\n",$seq_hash{$_},"\n";
}
close ID; system("rm tmp");
```

```
perl get-seq3.pl metagenemark_predictions.faa test-query.fa.cowrument.out.m9.hitid |
less
```

There are 2,547,270 fasta sequences

Loaded into memory, id as KEY, seq as VALUE

There are 104 seq ids, one id per line

Now the id is called to return the seq

Step 1: prepare the tabular format file (id + seq)
Step 2: load the tabular format file into memory as a hash
Step 3: read in the id file and extract seqs