# Solution to 1.1.Getting\_stared\_with\_perl

## Check and report version of PERL on your computer.

#Ans : open command prompt. Type perl -v

## Write a Perl program that prints your name and e-mail in the

# following format

#Name: Student\_name

#E-mail ID : student@uenf.br

#Ans: write the follwing code in a text file, save the file and run

**use** warnings**;**

**print** "Name: Kanhu charan"**;**

**print** "\n"**;**

# use escape character \ before @

**print** "E-mail ID : kanhu\@uenf.br"**;**

**print** "\n"**;**

## Write the same program as an one liner script.

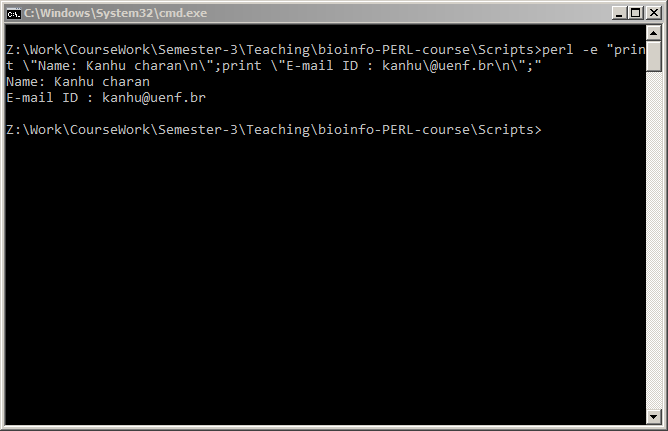
#Ans: On command prompt type the follwoing; and press Enter

# on Linux (use single quotes)

perl **-e** 'print "Name: Kanhu charan\n";print "E-mail ID : kanhu\@uenf.br\n";'

# on Windows (use double quotes and escape character)

perl **-e** "print \"Name: Kanhu charan\n\";print \"E-mail ID : kanhu\@uenf.br\n\";"



# Solutions to 5.Operations\_on\_variables

## Ask user to input a DNA sequence and print it in upper case letters.

#Ans:

**use** warnings**;**

**print** "Enter a DNA sequence: "**;**

$dna **=** <STDIN>**;**

$dna\_uc **=** **uc** $dna**;**

**print** "DNA in upper case : $dna\_uc\n"**;**

## Ask user to input a DNA sequence and print its first 3 nucleotides

# and last 3 nucleotides.

**use** warnings**;**

**print** "Enter a DNA sequence: "**;**

$dna **=** **uc** <STDIN>**;**

**chomp(**$dna**);**

$start\_3 **=** **substr(**$dna**,** 0**,**3**);**# position , length

$end\_3 **=** **substr(**$dna**,** **-**3**);** ## negative position

**print** "DNA seq: $dna\nFirst three: $start\_3\nLast three base: $end\_3\n"**;**

## You were provided with an array @DNA=(‘a’,’t’,’g’,’c’);

# Generate a random genetic code by combaining any 3 nucleotide bases.

# Write a program to generate and print a random amino acid

# translated from the genetic code.

# (Hint: assume have stored genetic codes in an hash %genetic\_code)

#Ans:

**use** warnings**;**

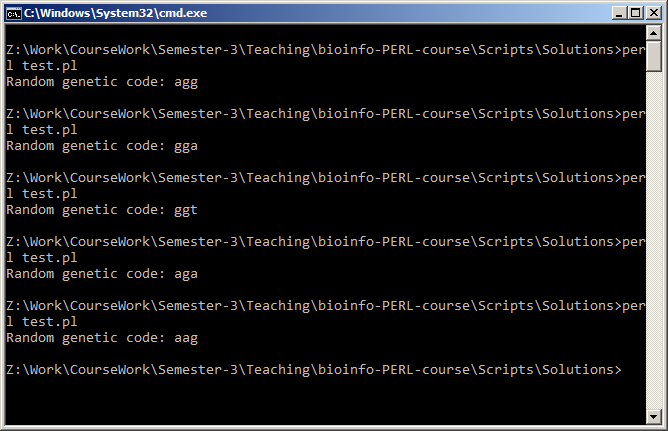
@DNA **=** **(**'a'**,**'t'**,**'g'**,**'c'**);**

$three\_base **=** $DNA**[int** **rand(**3**)].**$DNA**[int** **rand(**3**)].**$DNA**[int** **rand(**3**)];**

**print** "Random genetic code: $three\_base\n"**;**

# you may comment the follwoing code, as it will warn as error

**print** "Amino acid coded by $three\_base: $genetic\_code{$three\_base}\n"**;**



## Create an hash of 5 genes.

# Gene name as key and their lengths as values.

# Print the list of genes with increasing order of gene length.

**use** warnings**;**

%genes **=** **(**

ATP7B **=>** 345**,**

CYP86A4 **=>** 1022**,**

CER1 **=>** 654**,**

TPK1 **=>** 120**,**

MAN1 **=>** 876**,**

**);**

# numerically sorting by values

@genes\_or **=** **sort** **{**$genes**{**$a**}<=>** $genes**{**$b**}}** **keys** %genes**;**

**print** "Genes in ascending order of length\n"**;**

**print** "$genes\_or[0] : $genes{$genes\_or[0]}\n"**;**

**print** "$genes\_or[1] : $genes{$genes\_or[1]}\n"**;**

**print** "$genes\_or[2] : $genes{$genes\_or[2]}\n"**;**

**print** "$genes\_or[3] : $genes{$genes\_or[3]}\n"**;**

**print** "$genes\_or[4] : $genes{$genes\_or[4]}\n"**;**

## Use the same gene hash to display the gene list to the user and ask to type a gene name.

# Write a program to delete user defined gene from the hash.

# Print the updated gene list.

**use** warnings**;**

%genes **=** **(**

ATP7B **=>** 345**,**

CYP86A4 **=>** 1022**,**

CER1 **=>** 654**,**

TPK1 **=>** 120**,**

MAN1 **=>** 876**,**

**);**

@genes\_or **=** **keys** %genes**;**

**print** "Genes in the hash \n"**;**

**print** "$genes\_or[0] : $genes{$genes\_or[0]}\n"**;**

**print** "$genes\_or[1] : $genes{$genes\_or[1]}\n"**;**

**print** "$genes\_or[2] : $genes{$genes\_or[2]}\n"**;**

**print** "$genes\_or[3] : $genes{$genes\_or[3]}\n"**;**

**print** "$genes\_or[4] : $genes{$genes\_or[4]}\n"**;**

**print** "Please enter a gene name to delete:"**;**

**chomp(**$g **=** <STDIN>**);**

**delete** $genes**{**$g**};**

# get the keys

@genes\_or **=** **keys** %genes**;**

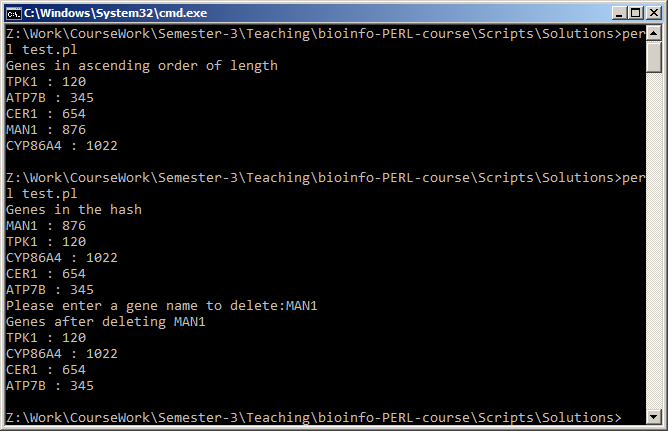
**print** "Genes after deleting $g\n"**;**

**print** "$genes\_or[0] : $genes{$genes\_or[0]}\n"**;**

**print** "$genes\_or[1] : $genes{$genes\_or[1]}\n"**;**

**print** "$genes\_or[2] : $genes{$genes\_or[2]}\n"**;**

**print** "$genes\_or[3] : $genes{$genes\_or[3]}\n"**;**



# # Solutions to 7.Perl\_file\_handling

## Write a Perl program that asks user to enter a file name and a target file name. Copy the content of the first file to the second one**.**

**use** warnings**;**

**print** "Please enter a file name to copy:"**;**

$file **=** <STDIN>**;**

**chomp(**$file**);**

**print** "Please enter a target file name:"**;**

$file\_o **=** <STDIN>**;**

**chomp(**$file\_o**);**

**open** IN**,** "< $file" **or** **die** "$!, $file"**;**

**open** COPY**,** "> $file\_o" **or** **die** "$!, $file\_o"**;**

**while** **(**$line**=**<IN>**){**

**print** COPY $line **;**

**}**

# close filehandles

**close** IN**;** **close** COPY**;**

**print** STDERR "$file copied to $file\_o successfully\n"**;**

#Write a Perl script to parse a BlastP tabular output file and save those

# hits with >45% identity.

**use** warnings**;**

$blast\_out **=** "blastp\_output.txt"**;**

**open** IN**,** "< $blast\_out" **or** **die** "$!, $blast\_out"**;**

**while** **(**$line**=**<IN>**){**

# split each line into an array

@record **=** **split** /\t/**,**$line**;**

**if(**$record**[**2**]>=**35**)** ## check the third column or the 2-index of the array >35

**{**

**print** "$line"**;**

**}**

**}**

# close filehandles

**close** IN**;**

## Write a Perl script that asks user to enter a Directory name. Print only the files present in the directory (excluding the directories**).**

**use** warnings**;**

**print** "Enter a directory name: "**;**

$dir **=** <STDIN>**;**

**chomp(**$dir**);**

**opendir** DIR**,** "$dir" **or** **die** "Error: $!, $dir\n"**;**

@ent **=** **readdir(**DIR**);**

**closedir** DIR**;**

**print** "Files present in $dir are : \n"**;**

**foreach** $f**(**@ent**){**

**print** "$dir\\$f\n" **if** **(-f** "$dir\\$f"**);**

**}**

## Open a FASTA file and copy its content to another file.

#Ans: Similar to copy files example

## Ask user to input to two file names and add '\_\_END\_OF\_FILE\_\_' to both of these files

#Ans:

**use** warnings**;**

**print** "Please enter first file name:"**;**

$file1 **=** <STDIN>**;**

**chomp(**$file1**);**

**print** "Please enter second file name:"**;**

$file2 **=** <STDIN>**;**

**chomp(**$file2**);**

**open** F1**,** ">> $file1" **or** **die** "$!, $file1"**;**

**open** F2**,** ">> $file2" **or** **die** "$!, $file2"**;**

**print** F1 "\_\_END\_OF\_FILE\_\_\n"**;**

**print** F2 "\_\_END\_OF\_FILE\_\_\n"**;**

# close filehandles

**close** F1**;** **close** F2**;**

## Open a GFF file and report total number of Genes present.

#Ans:

**use** warnings**;**

**print** "Please Enter GFF file name:"**;**

$gff **=** <STDIN>**;**

**chomp(**$gff**);**

$gene\_count **=** 0**;**

**open** GFF**,** "< $gff" **or** **die** "$!, $gff"**;**

**while(**$l**=**<GFF>**){**

@tabs **=** **split** /\t/**,**$l**;**

**if(**$tabs**[**2**]** **eq** 'gene'**){**

$gene\_count**++;**

**}**

**}**

# close filehandles

**close** GFF**;**

**print** "Total number of genes in $gff : $gene\_count\n"**;**

# Answer: for Athaliana.gff3: 27416

## Open a GFF file and report total number of Genes present per chromosome.

#Ans:

**use** warnings**;**

**print** "Please Enter GFF file name:"**;**

$gff **=** <STDIN>**;**

**chomp(**$gff**);**

%gene\_count**;** # create a hash to count for each Chromosome

**open** GFF**,** "< $gff" **or** **die** "$!, $gff"**;**

**while(**$l**=**<GFF>**){**

@tabs **=** **split** /\t/**,**$l**;**

**if(**$tabs**[**2**]** **eq** 'gene'**){**

$gene\_count**{**$tabs**[**0**]}++;** ## First column contains Chromsome name

**}**

**}**

# close filehandles

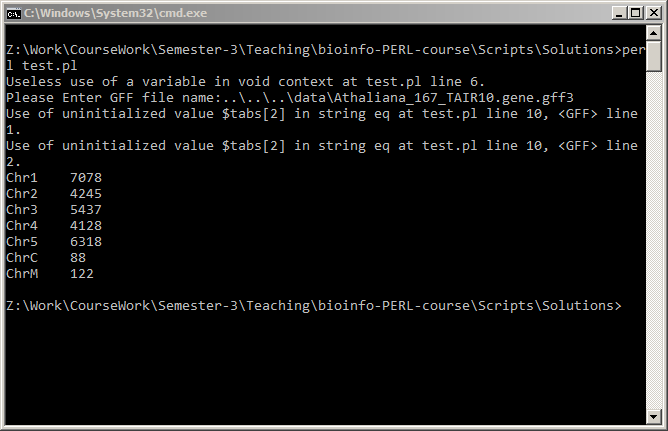
**close** GFF**;**

**foreach** $chr**(sort** **keys** %gene\_count**)**

**{**

**print** "$chr\t$gene\_count{$chr}\n"**;**

**}**



## Open a GFF file and report average legth of genes per chromosome.

**use** warnings**;**

**print** "Please Enter GFF file name:"**;**

$gff **=** <STDIN>**;**

**chomp(**$gff**);**

%gene\_count**;** # create a hash to count for each Chromosome

%gene\_length**;** # create a hash to total gene length for each Chromosome

**open** GFF**,** "< $gff" **or** **die** "$!, $gff"**;**

**while(**$l**=**<GFF>**){**

@tabs **=** **split** /\t/**,**$l**;**

**if(**$tabs**[**2**]** **eq** 'gene'**){**

$gene\_count**{**$tabs**[**0**]}++;** ## First column contains Chromsome name

$gene\_length**{**$tabs**[**0**]}+=**$tabs**[**4**]-**$tabs**[**3**]+**1**;** #4,5 column contains gene start and end locations

**}**

**}**

**close** GFF**;**

**foreach** $chr**(sort** **keys** %gene\_count**)**

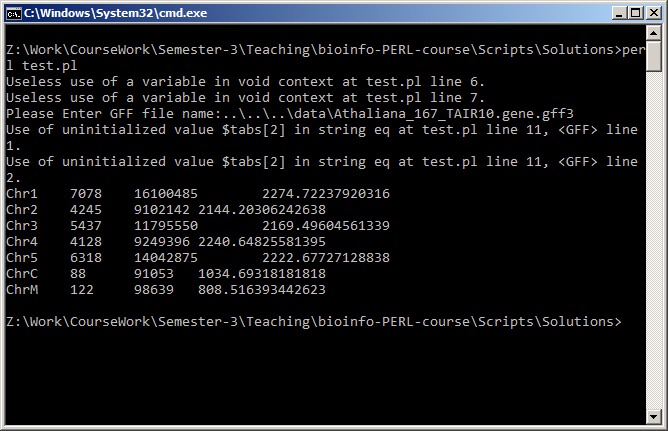
**{**

**print** "$chr\t$gene\_count{$chr}\t$gene\_length{$chr}\t"**;**

$avg **=** $gene\_length**{**$chr**}/**$gene\_count**{**$chr**};**

**print** "$avg\n"**;**

**}**



# Solutions to 10.Run\_external\_programs

## Write a script to perform blastP search at 10 different e-value

# cutoff. Report total number of hits predicted in each search.

#Ans:

**use** warnings**;**

**use** strict**;**

**my** @e\_values **=** **(**10**,**1**,**0.1**,**0.01**,**0.001**,**0.0001**,**0.00001**,**1e-10**,** 1e-15**,** 1e-30**);**

**my** $query **=** $ARGV**[**0**];**

**my** $db **=** $ARGV**[**1**];**

**my** %h**;** # to store no of hits for each e-val

**foreach** **my** $e**(**@e\_values**){**

**my** $cmd **=** "blastp -query $query -db $db -outfmt 6 -out blastp$e\.out -evalue $e"**;**

**print** STDERR "$cmd\n"**;**

**system(**$cmd**);**

**my** $hit **=** **`wc -l blastp**$e**\.out`;**

**chomp(**$hit**);**

$h**{**$e**}** **=** $hit**;**

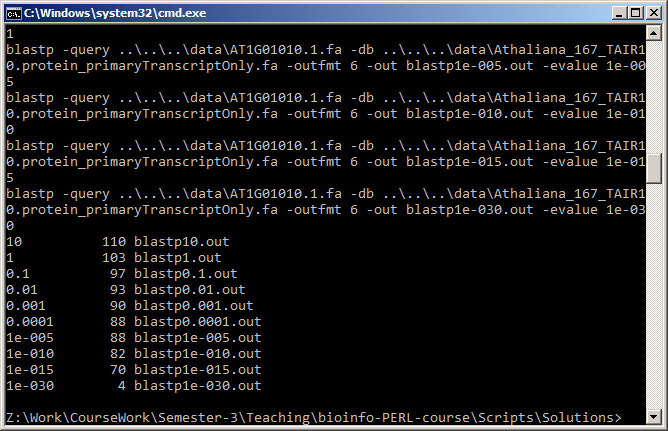
**unlink** **(**"$query\\_blastp.out"**);**

**}**

**foreach** **my** $e**(**@e\_values**){**

**print** "$e\t$h{$e}\n"**;**

**}**



## Write a script to automate BLASTp search, using each of the

# fasta files in the user defined directory against A.thaliana proteome.

#Ans:

**use** warnings**;**

**use** strict**;**

**my** $query **=** $ARGV**[**0**];** ## Enter directory containing fasta files

**my** $db **=** $ARGV**[**1**];** # database file path

**opendir** DIR**,** "$query" **or** **die** "Error: $! $query"**;**

**my** @files **=** **readdir(**DIR**);**

**closedir** DIR**;**

**foreach** **my** $f**(**@files**){**

**if(-f** "$query/$f" **and** $f**=~**m/\.faa$/**){** # check is it a \*.faa file

**my** $cmd **=** "blastp -query $query/$f -db $db -outfmt 6 -out blastp$f\.out"**;**

**print** "$cmd\n"**;**

**system(**"$cmd"**);**

**}**

**}**