Perl One-liner TP & quiz

TP - variables

❖ Write a one-liner Perl command that initializes a scalar variable with `"Hello world!"` and then display it

❖ Write a one-liner Perl command that initializes an array variable with three values : `"Hello"`, `"world"`, `"!"` and then display `"world"`

❖ Write a one-liner Perl command that initializes a hash variable with three keys/values: `"name"`:`"YOUR NAME"`, `"age"`:`"YOUR AGE"` and then display `"You are truly in the prime of life,YOUR NAME!"`

TP - variables - corrections

❖ Initialize a scalar variable with `"Hello world!"` and then display it

```
perl -e '$message = "Hello world!"; print "$message\n";'
```

❖ Initialize an array variable with three values : `"Hello"`, `"world"`, `"!"` and then display `"world"`

```
perl -e '@arr = ("Hello","world","!"); print "$arr[1]\n";'
```

❖ Initialize a hash variable with three keys/values : `"name"`:`"YOUR NAME"`, `"age"`:`"YOUR AGE"` and then display `"You are truly in the prime of life,YOUR NAME!"`

```
perl -e '%hash = ("name" => "NAME", "age" => "AGE"); print
"You are truly in the prime of life $hash{\"name\"}\n";'
```

TP - operators

❖ Calculate the price of an item during the sales. Complete the following line to return the price remaining to be paid after applying a discount of 10, and 40%

```
echo 275 | perl ...
```

❖ Count the number of lines in the fastq file (ERR.fastq) using `wc -l` and then use Perl to determine whether the file is valid (correct number of lines).

TP - operators - corrections

❖ Calculate the price of an item during the sales. Complete the following line to return the price remaining to be paid after applying a discount of 10, and 40%

```
echo 275 | perl -lne 'print "10% : " . ($_-($_*0.1)) . "\n40% : " . ($_-($_*0.4))'
```

❖ Count the number of lines in the fastq file (ERR.fastq) using `wc -l` and then use Perl to determine whether the file is valid (correct number of lines).

```
wc -l ERR.fastq | perl -lne '$v = ($_%4==0) ? "Valid" : "Invalid"; print $v;'
```

TP - functions

- Create an array that contains the first five natural numbers. Print the array. Create an new array shifting the elements left by one position (element 1 goes to 0) and setting the first element in the last position. Print the new array.

- Use the 3 tables below to print the favorite shoe color and size per each family member. Output lines should be in the format: `"Homer wears brown shoes size 12"`.

```
@family = ("Homer","Marge","Bart");
@shoe_color = ("Marge","blue","Bart","yellow", "Homer","brown");
@shoe_size = ("Bart",8,"Homer",12,"Marge",10)
```

TP - functions - corrections

- Create an array that contains the first five natural numbers. Print the array. Create an new array shifting the elements left by one position (element 1 goes to 0) and setting the first element in the last position. Print the new array.

```
perl -le '@n = (1..5); print join(" ", @n); @m = @n;
push(@m, shift(@m)); print join(" ", @m)';
```

- Use the 3 tables below to print the favorite shoe color and size per each family member. Output lines should be in the format: `"Homer wears brown shoes size 12"`.

```
perl -le '@family = ("Homer","Marge","Bart"); @shoe_color = ("Marge","blue","Bart","yellow",
"Homer","brown"); @shoe_size = ("Bart",8,"Homer",12,"Marge",10); %color = @shoe_color; %size
= @shoe_size; print join("\n", "$family[0] wears $color{$family[0]} shoes size
$size{$family[0]}","$family[1] wears $color{$family[1]} shoes size $size{$family[1]}",
"$family[2] wears $color{$family[2]} shoes size $size{$family[2]}")'
```

TP - conditional statement

- Using the `samples.tsv` file and knowing the size of the genome (2,922,600,443 bp), display the number of samples with coverage <10X, between 10 and 50X and >50X

- How was the header taken into account and why?

TP - conditional statement - corrections

❖ Using the `samples.tsv` file and knowing the size of the genome (2,922,600,443 bp), display the number of samples with coverage <10X, between 10 and 50X and >50X

```
perl -lne '@l=split(/\t/); $x=$l[6]/2922600443; if($x<10)
{$a++;} else { if($x<50) {$b++;} else {$c++}} print "$a\t
$b\t$c"' samples.tsv | tail -n1
```

❖ How was the header taken into account and why?

```
perl -le 'print "string"/2922600443;' # prints "0"
```

TP - functions and structures

- Read the `samples.tsv` file, calculate the read length per line with `perl` and use the `sort` and `uniq` shell commands to get the number of samples with the same read length.

- Same exercise but without using `sort` and `uniq`.

- Read the `samples.tsv` file, calculate the average number of reads and the average number of bases between all samples.

TP - functions and structures - corrections

- Read the `samples.tsv` file, calculate the read length per line with `perl` and use the `sort` and `uniq` shell commands to get the number of samples with the same read length.

```
cat samples.tsv | perl -lne 'next if $.==1; @l=split(/\t/); print int($l[6]/$l[5]) if
$l[5]' | sort -n | uniq -c
```

- Same exercise but without using `sort` and `uniq`.

```
cat samples.tsv | perl -lne 'next if $.==1; @l=split(/\t/); $h{int($l[6]/$l[5])}++ if
$l[5]; END{foreach (keys %h){print "$_ $h{$_}"}}'
```

- Read the `samples.tsv` file, calculate the average number of reads and the average number of bases between all samples.

```
cat samples.tsv | perl -lne 'next if $.==1; @l=split(/\t/); $n++; $r+=$l[5];
$b+=$l[6];END{print "Average #reads: ".int($r/$n); print "Average #bases:
".int($b/$n)}'
```

Quiz perl command options

https://digistorm.app/p/8191099

Question 1

The `-e` option is used to enter a program line, so Perl does not look for a file name to execute. What will be displayed when the following line is executed?

```
$ echo Hello World | perl -e 'print $_'
```

Answer

The `-e` option is used to enter a program line, so Perl does not look for a file name to execute. What will be displayed when the following line is executed?

```
$ echo Hello World | perl -e 'print $_'

$
```

Question 2

The `-l` option is used to

- ❖ remove the record separator on input
- ❖ add the record separator to all `print` instructions on output

What will be displayed when the following line is executed?

```
$ echo Hello World | perl -le 'print $_'
```

Answer

The `-l` option is used to

❖ remove the record separator on input
❖ add the record separator to all `print` instructions on output

What will be displayed when the following line is executed?

```
$ echo Hello World | perl -le 'print $_'



$
```

Question 3

The -n option is used to enclose your program in a loop of the following type

```
while (<STDIN>) { my_program }
```

What will be displayed when the following line is executed?

```
$ echo Hello World | perl -lne 'print $_'
```

Answer

The `-n` option is used to enclose your program in a loop of the following type

```
while (<STDIN>) { my_program }
```

What will be displayed when the following line is executed?

```
$ echo Hello World | perl -lne 'print $_'

Hello World
```

Question 4

The `-p` option is used to enclose your program in a `while` loop, like the `-n` option, and display the lines automatically. What will be displayed when the following line is executed?

```
$ echo Hello World | perl -lpe 's/l/m/g'
```

Answer

The `-p` option is used to enclose your program in a `while` loop, like the `-n` option, and display the lines automatically. What will be displayed when the following line is executed?

```
$ echo Hello World | perl -lpe 's/l/m/g'

Hemmo Wormd
```

Question 5

The `-a` option is used to enable the auto-split mode when used with `-n` or `-p` and thus an implicit `split` command to the `@F` array is done at the start of the `while` loop. What will be displayed when the following line is executed?

```
$ echo Hello World | perl -lane 'print $F[1]'
```

Quiz Perl regex

https://digistorm.app/p/8883214

Memo

❖ `^` `$`	- match at the beginning or at the end of the string
❖ `|`	- alternation metacharacter (OR)
❖ `[a-z]`	- defining a character class
❖ `*`	- match zero or more times
❖ `+`	- match at least one time
❖ `[^a-z]`	- negating a character class

Answers

1. /^P/	**P**erl
2. /Perl|Python/	**Perl**
3. /[a-z]1/	ab**c1**23
4. /[a-z]2/	abc123
5. /a+b/	**aab**b

6. /b*a/	**a**abb
7. /a*b/	a* **b**
8. /[^0-9]/	**a**bc
9. /[^a]bc/	bc
10. /[^b]*c/	ab**c**

TP - regex

- ❖ Write a regular expression to check whether a DNA sequence begins with `ATG` and ends with `TAA`, `TAG` or `TGA`

- ❖ Write a regular expression to check the validity of an email address

TP - regex - corrections

- Write a regular expression to check whether a DNA sequence begins with ATG and ends with TAA, TAG or TGA

```
echo ATGAGTGAGTAGTAGTTAAATTAG | perl -lne 'print "is CDS" if
/^ATG([ATGCN][ATGCN][ATGCN])+(TAA|TAG|TGA)$/i'
```

- Write a regular expression to check the validity of an email address

```
echo jean.saisrien@dutout.fr | perl -lne 'print "valid" if
/^[a-z0-9.-]+@[a-z0-9.-]+\.[a-z]{2,3}$/'
```

TP - Build cmd files to run on a cluster

- Write a Perl one-liner that generates a bwa command file from the `samples.tsv` file (e.g. `bwa mem REF.fa ERR3281353_1.fastq.gz ERR3281353_2.fastq.gz | samtools sort - > ERR3281353.bam`)

TP - Build cmd files to run on a cluster - corrections

- Write a Perl one-liner that generates a bwa command file from the `samples.tsv` file (e.g. `bwa mem REF.fa ERR3281353_1.fastq.gz ERR3281353_2.fastq.gz | samtools sort - > ERR3281353.bam`)

```
perl -F'\t' -lane 'next if(/^st/); @fq=split(";",$F[8]);
$fq[0]=~s/^ftp.*\///; $fq[1]=~s/^ftp.*\///; print "bwa mem
REF.fa $fq[0] $fq[1] | samtools sort - > $F[3].bam"' samples.tsv
```

TP - Stat on GFF

- ❖ Write a Perl one-liner that counts and displays the number of genes for each biotype in the GFF file

TP - Stat on GFF - corrections

❖ Write a Perl one-liner that counts and displays the number of genes for each biotype in the GFF file

```
perl -F'\t' -lane 'if($F[8]=~/gene_biotype=(.*?);/) { $h{$1}++;
}  END { foreach my $k (sort keys %h) { print "$k\t".$h{$k}; }
}' file.gff
```

TP - Stat on GFF

- Write a Perl one-liner which, per chromosome, counts the total number of genes, the number of genes on each strand, and calculates the average length of these genes in the GFF file

TP - Stat on GFF - corrections

❖ Write a Perl one-liner which, per chromosome, counts the total number of genes, the number of genes on each strand, and calculates the average length of these genes in the GFF file

```
perl -F'\t' -lane ' $h{$F[0]}{"nb"}++; $h{$F[0]}{"len"}+=($F[4]-$F[3]+1);
($F[6] eq "+") ? $h{$F[0]}{"+"}++ : $h{$F[0]}{"-"}++; END { print
"#Chromosome\tNbGene\tNbGene+\tNbGene-\tMeanLen"; foreach my $k (sort keys
%h) { print "$k\t" . $h{$k}{"nb"} . "\t" . $h{$k}{"+"} . "\t" . $h{$k}{"-"} .
"\t" . int($h{$k}{"len"}/$h{$k}{"nb"}); } }' file.gff
```

TP - Correspondence table

- Write a Perl one-liner which adds the column `sample_alias` of the file `sample_names.tsv` to the file `samples.tsv`.

- Write a Perl one-liner that replaces the `sample_accesssion` column in the `samples.tsv` file with the `sample_alias` column in the `sample_names.tsv` file.

TP - Correspondence table - corrections

- Write a Perl one-liner which adds the column `sample_alias` of the file `sample_names.tsv` to the file `samples.tsv`

```
cat sample_names.tsv samples.tsv | perl -lane 'if(scalar(@F) == 2) {
$h{$F[0]}=$F[1]; } else { print "$_\t$h{$F[1]}"; }'
```

- Write a Perl one-liner that replaces the `sample_accesssion` column in the `samples.tsv` file with the `sample_alias` column in the `sample_names.tsv` file.

```
cat sample_names.tsv samples.tsv | perl -lane 'if(scalar(@F) == 2) {
$h{$F[0]}=$F[1]; } else { $F[1] =~ s/$F[1]/$h{$F[1]}/; print
join("\t",@F); }
```