PERL QUESTIONS

1) What arguments do you frequently use for Perl Interpreter and what do they mean?

`perl –cw`: - It is used to check the syntax and warning message.

2) What is the difference between single - quoted text and double - quoted text?

Single quotation marks do not interpret, and double quotation marks do

3) How to find last index value of an array?

$#Array_Name **OR** $Array_Name [-1] (2nd - better);

$#Array_Name it will return last index of array but $Array_Name [-1] will return last element of array not last index value

4) What is the difference between chop and chomp functions?

`chop` - removes the last character and returns the character chopped

`chomp` **-** removes any trailing string that corresponds to the current value of $/ and returns number of characters removed

5) What do the symbols $ @ and % mean when prefixing a variable?

$ - Indicates scalar data type

@ - Indicates an array

% - Indicates an hash or an associative array

6) What elements of the Perl language could you use to structure your code to allow for maximum re-use and maximum readability?

subroutines and modules – Element “sub” ,“use” ,“require”, “package”

7) Write syntax for replace a char in string and syntax to store the number of replacements?

```
$str='Hello';
$cnt = ($str =~ s/l/i/g);
print $cnt;
```

OR

```
#!usr/bin/perl
use strict;
use warnings;
my $string="XmanXseriesx”;
my $count = ($string =~ tr/X//);
print "There are $count Xs in the string\n";
print $string;
```

8) Explain difference between ‘*use*’ , ‘*require*’ and do( ).

- `use()` - It included objects are verified at the time of compilation.
- `require`() – It included objects are verified at the run time. Before attempting to load a file or a module with use() or require(), Perl checks whether it's already in the %INC hash. If it's there, the loading and therefore the compilation are not performed at all
- `do`() - do() does unconditional loading--no lookup in the %INC hash is made

9) How to implement stack in perl?

Stack is LIFO (Last in First out) In perl that could be implemented using the push () and pop () functions. Push () adds the element at the last of array and pop () removes from the end of an array.

10) What is `Grep` used for in Perl?

select certain elements from the array or list

11) What does '`qw()`' mean? What's the use of it?

qw stands for 'Quote word'. It usually used to create array.

qw// is a construct, which quotes words delimited by spaces.

Ex. `@arr qw/one two three/;`

Each space considered as separator for new element

12) How do we know how many entries are in a hash?

```
$num_keys = scalar keys %hash;
my $key_count = keys %hash; # must be scalar context!
my $defined_value_count = grep { defined } values %hash;
my $vowel_count = grep { /[aeiou]/ } keys %hash;
my @defined_values = grep { defined } values %hash; # list of
matching
```

13) How do you know the reference of a variable whether it is a reference, scalar, array, or hash?

Using ref function ref($x) eq 'ARRAY'

14) How to concatenate strings in Perl?

```
$name = 'checkbook';
$filename = join "", "/tmp/", $name, ".tmp"; OR
$filename = "/tmp/" . $name . ".tmp"; OR
my $var=hello; $var .= World;
```

15) Explain about some quantifiers in regular expressions?

Greedy	Non greedy	Allowed numbers for a match
?	??	0 or 1 time
+	+?	1 or more times
*	*?	0 or more times
{i}	{i}?	exactly i times
{i,}	{i,}?	i or more times
{i,j}	{i,j}?	at least i times and at most j times

16) Write a regular expression to match floating-point number with optional sign?

`^[-+]?\d*\.?\d*$`

17) Write a regular expression to match an email id?

My $email = hi@kesavan.info ;

$email =~ /^(\w+|\d+)(_|.)?(\w+|\d+)[@](\w+).(in|net|co(m?|\.?i?n?))$/;

`[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+(?:[A-]{2}|com|org|net|gov|mil|biz|info|mobi|name|aero|jobs|museum)\b`

18) What is the difference between function and subroutine?

Both are same in perl

Normal def:

A function returns a value whereas a subroutine does not.

A function should not change the values of actual arguments whereas a subroutine could change them.

19) Does Perl have a round() function Give an example?
Perl does not have an explicit round function. Ex:

```
sub round {
     my($number) = shift;
     return int($number + .5);
}
          OR
return int($number + .5 * ($number <=> 0));
```

20)What is variable suicide?
Variable suicide is when you (temporarily or permanently) lose the value of a variable. It is caused by scoping through my() and local() interacting with either closures or aliased foreach() interator variables and subroutine arguments.

```
     my $f = "foo”;
     sub T {
          while ($i++ < 3) { my $f = $f; $f .= "bar"; print
     $f, "\n"}

     T;
     print "Finally $f\n”;
```

21) Explain what strings will match the following patterns

```
 i. /ice\s*cream/    matches the string with/without white space.e.g.:
IceCream/Ice Cream/Ice                  Cream
ii. /\d\d\d/      matches 3 consecutive digits in a string. e.g.:
222/Ice222cream/233Ice/cream223
iii. /^\d+$/        matches the string only with digits.          e.g.:
23434/1/4234/55
iv. /ab?c[,.:]d/    matches the string with ‘a’ followed by with/without
‘b’ and with ‘c’ and anyone of ‘,.:’ and with ‘d’. e.g.:
iceabc.dcream/iceac,dcream/ac:d/
v. /xy|yz+/         matches string with xy or yz e.g.:
icexycream/iceyzcream/
vi. /[\d\s]{2,3}/   matches 2 to 3 digits/white spaces e.g.: ice
33/3333ice/    ice/
vii. /"[^"]"/       matches a single character in a string except “ within
double quotes. e.g.: ice”c”cream/ice”,”cream
```

22. What will be the value of $1 after execution of the following code?

```
   my $txt = 'I am learning Perl';  $txt =~ /(\w+)$/;
```

Perl

23. What will be the value of $str after execution of the following code?

```
           my $str = '112133'; $str =~ s/(.)\1/$1/g;
           1213
```

26.what is the difference push,pop,shift and unshift
27.what isthe difference between structured language and object oriented language
28.Explain about function override and overload

overloading

the same function name and different parameters is called overloading.

```
function ss(int a,int b)
function ss(int a,int b,int d)
```

overriding.

The base class method is override by the derived class.

```
class a{
     private int dd(){
          console.writeline("hai")
     }
}

class b inherits a{
     private int dd(){
          console writeline("hai everybody");
     }
}

the output is
hai everybody
```

29.What is advantage of overriding
30.How to create Package
31.Can we rename new() fuction to another name function for creating the object for package
32.List out the used packages
33.How to connect the DB using perl
34.What is the ment by cron tab and crontab -r ?
35.How to implement stack in perl?
36. list out the unix commands
37. grep
38.$#array_name what it will returns?
41.What are the keys and values of INC hash
42.How to get last 10 lines of file
43.How u open the file and what are the modes
44.What is the difference between do and use,require
45.while connecting db the word mysql is case centivie?ANS: Yes
46.how to display list of process in unix?ANS:using top command
47.what is the use of topANS:to list of all the running process in unix server
48.how to kill the process unixANS:Kill process id
49.How to remove a directory using perl?
50.What is ment by .profile file in unix
.profile is a Unix shell script which is executed for a user every time that user logs in.
A standard .profile will set important environment variables such as $PATH and may also run commands, such as `fortune`.
The .profile file will be stored in the users home directory.
.profile is used by the Bourne, Korn, and Bash shells. The C and TCSH shells use startup files called .login and .cshrc instead.
In addition to .profile, the Korn shell also uses a startup file called .kshrc.
Also, the /etc/profile shared startup file will by executed by all Bourne, Korn, and Bash shells
51.how to find a perticular word in file ?ANS:using grep command
52.How u run the unix commands in perl ANS:by using system

53.what functions we have under ftp module ANS: put,mput,get,mget

%INC hash :

The key is the name of the file or module
The value is the full path to it in the file system.

Crontab Commands

You can execute crontab if your name appears in the file /usr/lib/cron/cron.allow.
If that file does not exist, you can use crontab if your name does not appear in the file /usr/lib/cron/cron.deny.

```
crontab -e     Edit your crontab file, or create one if it
doesn't already exist.
crontab -l      Display your crontab file.
crontab -r      Remove your crontab file.
crontab -v      Display the last time you edited your crontab
file. (This option is only available on a few systems.)
```

A crontab file has five fields for specifying day , date and time followed by the command to be run at that interval.

```
*     *    *    *    *  command to be executed
-     -    -    -    -
|     |     |     |    |
|     |     |     |    +------ day of week (0 - 6)
(Sunday=0)
|     |     |     +----------- month (1 - 12)
|     |     +----------------- day of month (1 - 31)
|     +----------------------- hour (0 - 23)
+--------------------------- min (0 - 59)
```

* in the value field above means all legal values as in braces for that column.
The value column can have a * or a list of elements separated by commas. An element is either a number in the ranges shown above or two numbers in the range separated by a hyphen (meaning an inclusive range).
Note: The specification of days can be made in two fields: month day and weekday. If both are specified in an entry, they are cumulative meaning both of the entries will get executed .

ps and top in unix. The **ps** command displays active processes. options:

-a	Displays all processes on a terminal, with the exception of group leaders.
-c	Displays scheduler data.
-d	Displays all processes with the exception of session leaders.
-e	Displays all processes.
-f	Displays a full listing.
-g*list*	Displays data for the *list* of group leader IDs.
-j	Displays the process group ID and session ID.
-l	Displays a long listing

-p*list*	Displays data for the *list* of process IDs.
-s*list*	Displays data for the *list* of session leader IDs.
-t*list*	Displays data for the *list* of terminals.
-u*list*	Displays data for the *list* of usernames.

About top - Display Linux tasks. Syntax

```
top -hv | -bcisS -d delay -n iterations -p pid [, pid ...]
```

IP ADDRESS MATCH :

/^([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])$/

Note : Add any other if you aware of ip address match . Its frequently asked in interviews

Nice Command in Unix:

Invokes a command with an altered scheduling priority
Ex: **nice +13 pico myfile.txt** - runs the pico command on myfile.txt with an increment of +13.
++

1) What is the output of the following perl program ?

```
$var1 = "program.java";
$var1 =~ s/(.*)\.java/$1.perl/;
print $var1;
```

O/P => program.perl

2) Can you write a short Perl script to generate a list of all prime numbers between 1 and 100?

```
for ( $i = 1; $i <= 100 ; $i++ ){

    if ( $i ==1 || $i == 2 ){
    print "Prime Number $i \n";
    }
    else{
        $j = $i / 2 ;
        $k = 3 ;
        $flag = 0;
        while( !$flag ){
            if ( $i % $k == 0 ){
            $flag =1 ;
            }
            else{
                $k += 2 ;
                if ( $k > $j ) {
                print " $i is prime \n";
                $flag = 1;
            }
        }
    }
}
}
```

OTHER OPTIONS:

```
######################
$i=3;
while($i<=100)
{
   $j=2;
   $flag =0;
   while($j<=($i/2))
   {
      if ($i%$j == 0)
      {
         $flag = 1;
         last;
      }
      $j++;
   }
   if ( $flag == 0)
   {
      #print "PRIME<$i>\n";
   }
   $i++;
}

#########################
my @arr = 1..100;
foreach $i ( @arr )
{
   my @match =();
   foreach $j ( @arr )
   {
      if ( int($i/$j)==($i/$j)  )
      {
         push @match,$j;
      }
   }
   print "Prime:$i\n" if ( $#match == 1 );
}
```

+++

3) You are given a plain text file containing many lines of the form “name=value” for a numeric value, e.g.

Bob=817
Sally=1420
John=817
Colleen=22
Richard=456

Write a script to process this text file and produce as output a list of all of the names within it, sorted in descending order by the magnitude of the value for each. You may assume that the names are unique

```
open ( FOPEN ,"file");
while(<FOPEN>){
chomp($_);
my ($A,$B) = (split(/\=/,$_)) [0,1];
```

```
$C{$B} = $A ;
}
close(FOPEN);
foreach my $val ( sort { $b <=> $a } keys %C ){
    print " Values is $C{$val}\n";
}
```

+++
+++
++++++++++++++++++++++++

4)

```
sub foo {
my $word = shift;
return sub {my $otherWord = shift;
print "$word $otherWord\n";};
}

sub foo {
my $word = shift;
return sub {my $otherWord = shift;
print "$word $otherWord\n";};
}

my $sub = foo("One","Two");
&$sub("Three","Four");
```

What is printed when the above code is executed?
Choice 1
One One
Choice 2
One Two
Choice 3
One Three
Choice 4
One Four
Choice 5
Three Four

O/P Choice - 3
+++
+++
++++++++++++++++++++++

5) Suppose $x contains a number. Which one of the following statements sets $y to be a string containing the octal value of $x?

Choice 1
`$y = oct($x);`
Choice 2
`$y = octal($x);`
Choice 3
`$y = itoa($x, 8);`

```
Choice 4
sprintf($y, "%o", $x);
Choice 5
$y = sprintf("%o", $x);
O/P Choice -5
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++
```

6) We use a Perl app model called HTML::Mason. A Mason "component" is just a text file, containing Perl code and other special syntax embedded within normal HTML. A simple example component is:

```
---------------------------------------
<html>
     <body> The value is: <% $value %> </body>
</html>
<%doc>
A basic component to display the value of a variable, $value
</%doc>
<%init>
my $value = "apple";
</%init>
---------------------------------------
```

The "<%doc>….</%doc>" and "<%init>….</%init>" sections describe "blocks" within the component.

If this example is contained in a file named "sample.cmp", write a simple command-line Perl script to be invoked as "perl yourscript.pl sample.cmp", which extracts the "doc" block and prints its contents to the screen.

O/P =>

```
open ( FOPEN , "$ARGV[0]" );

my $flag = 0 ;
while ( <FOPEN> ) {

if ( $_ =~ /\<\%doc\>/ ) {

$flag = 1 ;
}
if ( $flag == 1 ) {

print "Line $_ \n";
}
if( $_ =~ /\<\/\%doc\>/ ) {

$flag = 0 ;
}
}
close( FOPEN );
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

7) Have List of numbers in comma separated . Eg (1,2,3,4) In that i need the 3rd Value from the list using unix commands as well as in AWK

1) UNIX : echo 1,2,3,4 | cut -d "," -f3
2) AWK : echo 1,2,3,4 | awk -F, '{ print $3 }'

8) give information about **"nohup"** unix command

`nohup' runs the given COMMAND with hangup signals ignored, so that the
command can continue running in the background after you log out

Example: **nohup find -name '*' -size +1000k > log.txt**

9) difference between command "df" and "du"
df => Will give the Unix server Device space Occupy files and foders
du => Will give the Size of input files or folder interms of bytes

10) a-z unix commands
11) difference between "diff" and "cmp" command in UNIX ?

"cmp" => Will give the **first** difference between the files (information will be Line Number and Character position)

"diff" => Will the Show the **all** the difference between in the files

---

1. File1.txt
Monthly,6700
Monthly,6700
Yearly,6700
Monthly,6700
Yearly,6700
Monthly,6700
Use awk command and print the total of column 2 for Monthly.

2.
* File1 contains billion records and File2 contains billion records.
* Both the column of the file should be same
* check the first column of the file1 should match with third column of the file2
Tell me the best scenario to do this?
3. Pass the array value and print the values based on the ginen output
4. Input date: "2009-12-12 18:30:30" should be shown as "1234562334"

awk 'NR==FNR{A[$1$2]=$4;next} $5=A[$2$3]' OFS="\t" file2 file1
awk -F, 'NR==FNR{a[$1$2]=$3;next}a[$1$2]{$4=a[$1$2];print}' OFS="," file1 file2

awk 'NR==FNR{A[$1$2]=$4;next} $5=A[$2$3]' OFS="\t" file2 file1

---

exec executes a command and never returns. It's like a return statement in a function. If the command is not found execute returns false. It never returns true, because if the command is found it never returns at all. There is also no point in returning STDOUT, STDERR or exit status of the command. You can find documentation about it in perlfunc, because it is a function.
system executes a command and your perl script is continued after the command has finished. The return value is the exit status of the command. You can find documentation about it in perlfunc.
backticks like system executes a command and your perl script is continued after the command has finished. In contrary to system the return value is STDOUT of the command. qx// is equivalent to backticks. You can find documentation about it in perlop, because unlike system and execit is an operator.
Other ways What is missing from the above is a way to execute a command asynchronously. That means your perl script and your command run simultaneously. This can be accomplished with open. It allows you to read STDOUT/STDERR and write to STDIN of your command. It is platform dependent though.

There are also several modules which can ease this tasks. There is IPC::Open2 and IPC::Open3 and IPC::Run, as well as Win32::Process::Create if you are on windows.

- ***exec*** replaces the current process with another one.
- ***system*** runs another program, wait for its completion.
- ***fork*** copies the current process; both the original and the copy continue from the same point.
- ***pipe*** sets up a pipe between two handles.
- ***syscall*** makes a low-level system call.
- ***eval*** executes (the string form will first compile) a piece of Perl code

---

exec
- exec is used to execute the given process by replacing the current process.
- If the given process get executed successfully then exec will not return the value.
- exec returns the value in case of failure only.

system
- System is also doing the same thing as exec but system returns value in both success and failure cases.
- And parent process waits for the child process to complete.
- System() runs the command through a shell,while exec() runs the command directly.

fork
- fork is used to create a new process(child process).
- And it is returning the PID of child to parent and zero to child if the fork is successful.
- The difference between the fork and exec is exec replaces the current process but fork

doesn't.

pipe
- pipe is used for communicating between two processes.
- We can use both named and nameless pipes.
- It returns open a pair of pipes.
- In one end we can write.
- And in another end we can read the content.

syscall
- syscall is used to call the system call which is specified as a first argument.
- Remaining elements are the arguments to the system call.

> die is used to throw an exception (catchable using eval).
> exit is used to exit the process.
> die will set the error code based on $! or $? if the exception is uncaught.
> exit will set the error code based on its argument.
> And of course,
> die outputs a message
> exit does not.
> **Update**: Added "catchable" bit by request of ysth.

Export allows to export the functions and variables of modules to user's namespace using the standard import method. This way, we don't need to create the objects for the modules to access it's members.

@EXPORT and @EXPORT_OK are the two main variables used during export operation.
@EXPORT contains list of symbols (subroutines and variables) of the module to be exported into the caller namespace.
@EXPORT_OK does export of symbols on demand basis.

Sample program

Let us use the following sample program to understand Perl exporter.

```
package Arithmetic;

use Exporter;

@ISA = qw(Exporter); # base class of this(Arithmetic) module
@EXPORT = qw(add subtract); # Exporting the add and subtract
routine
# Exporting the multiply and divide routine on demand basis.
@EXPORT_OK = qw(multiply divide);

sub add{
my ($no1,$no2) = @_;
```

```
my $result;
$result = $no1+$no2;
return $result;
}

sub subtract{
my ($no1,$no2) = @_;
my $result;
$result = $no1-$no2;
return $result;

}

sub multiply{
my ($no1,$no2) = @_;
my $result;
$result = $no1*$no2;
return $result;
}

sub divide{
my ($no1,$no2) = @_;
my $result;
$result = $no1/$no2;
return $result;

}
```

In the above example, we defined the arithmetic module with four functions. By default add() and subtract() functions are exported to the user's/caller's namespace.
"use Arithmetic" statement imports the subroutines from Arithmetic module that are exported by default.
"use Arithmetic qw(multiply divide)" indicates that these two routines gets exported only when it is specifically requested as shown in the following code snippet.

```
#! /usr/bin/perl

use strict;
use warnings;

use Arithmetic;
use Arithmetic qw(multiply divide);

print add(1,2),"\n";
print multiply(1,2),"\n";
```

As we seen above, in the main program we used the Arithmetic module with default import (add and subtract) and on-demand import (multiply and divide).

How Imports Gets Done

Without Exporter, to import functions from Arithmetic module into the main package, we can write a code-snippet like the following to import some of the functions.

```
sub import {
no strict 'refs';
for (qw(add subtract multiply )) {
*{"main::$_"} = \&$_;
}
}
```

The code in the main package/program are as follows,

```
#!/usr/bin/perl
use Arithmetic; # import() subroutine in Arithmetic module
gets invoked automatically.
print add(1,2);
```

In the above code-snippet, from the current package (Arithmetic), these routines (add, subtract, multiply) gets imported into the main package. This provides limited functionality. i.e In case if the use is invoked from packages other than main, this won't work properly. But in the Perl Exporter module, we don't have this limitation.