Date:18/10/2021

**Fall Semester 2021-2022**

# ECE5030 - Scripting Languages for VLSI Design Automation

**M.Tech VLSI Design**

**School of Electronics Engineering**

**Vellore Institute of Technology**

**Name: Shreyas S Bagi**

**Register Number: 21MVD0086**

**Theory Slot: E2**

**Date:18/10/2021**

# Digital Assignment 1

**Aim:** Write PERL script, execute and check for errors and correct them

## Question 1 :

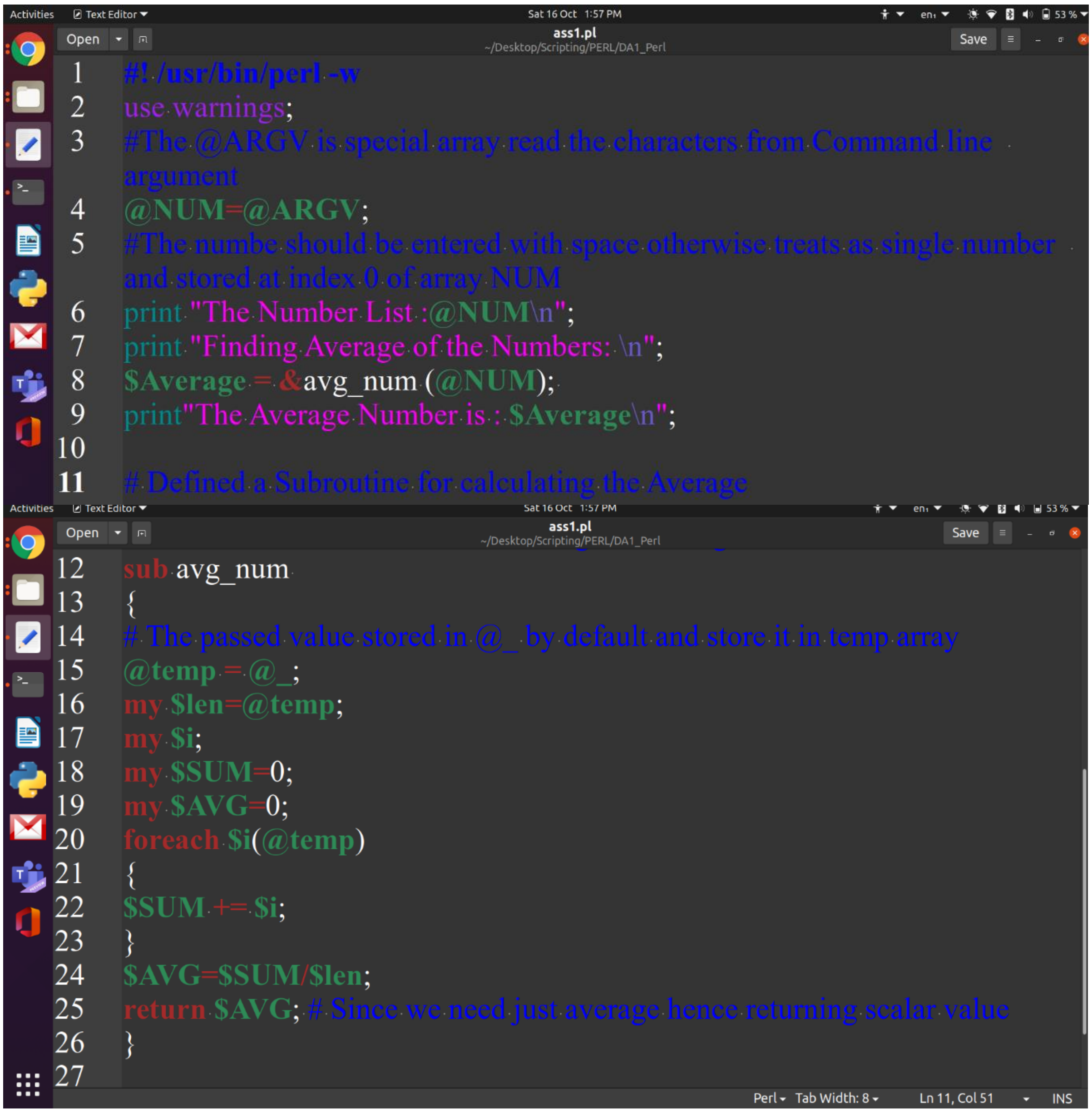
Write a Perl script to find the average of ***n*** numbers. The numbers are passed to the perl script as command line arguments while executing the script.

## Program/ Code:

```
#! /usr/bin/perl -w
use warnings;
#The @ARGV is special array read the characters from Command line argument
@NUM=@ARGV;
#The numbe should be entered with space otherwise treats as single number and stored at
index 0 of array NUM
print "The Number List :@NUM\n";
print "Finding Average of the Numbers: \n";
$Average = &avg_num (@NUM);
print"The Average Number is : $Average\n";

# Defined a Subroutine for calculating the Average
sub avg_num
{
# The passed value stored in @_ by default and store it in temp array
@temp = @_;
my $len=@temp;
my $i;
my $SUM=0;
my $AVG=0;
foreach $i(@temp)
{
$SUM += $i;
}
$AVG=$SUM/$len;
return $AVG; # Since we need just average hence returning scalar value
}
```

Date:18/10/2021

## PERL Code Screenshots:

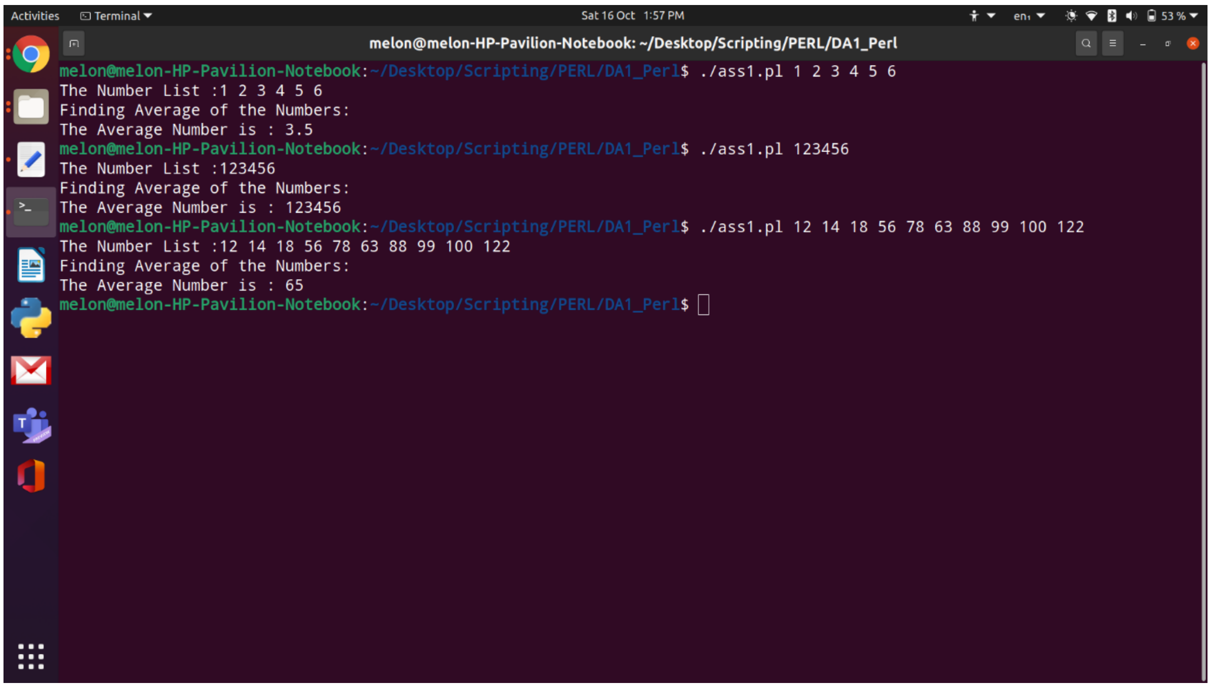


```perl
#! /usr/bin/perl -w
use warnings;
#The @ARGV is special array read the characters from Command line argument
@NUM=@ARGV;
#The numbe should be entered with space otherwise treats as single number and stored at index 0 of array NUM
print "The Number List :@NUM\n";
print "Finding Average of the Numbers: \n";
$Average = &avg_num (@NUM);
print"The Average Number is : $Average\n";

# Defined a Subroutine for calculating the Average
sub avg_num
{
# The passed value stored in @_ by default and store it in temp array
@temp = @_;
my $len=@temp;
my $i;
my $SUM=0;
my $AVG=0;
foreach $i(@temp)
{
$SUM += $i;
}
$AVG=$SUM/$len;
return $AVG; # Since we need just average hence returning scalar value
}
```

**Figure 1.1** Screenshot of PERL code for calculation of average of numbers.

Date:18/10/2021

## Output Screenshot:


Activities
Terminal
Sat 16 Oct 1:57 PM
melon@melon-HP-Pavilion-Notebook: ~/Desktop/Scripting/PERL/DA1_Perl

```
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ ./ass1.pl 1 2 3 4 5 6
The Number List :1 2 3 4 5 6
Finding Average of the Numbers:
The Average Number is : 3.5
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ ./ass1.pl 123456
The Number List :123456
Finding Average of the Numbers:
The Average Number is : 123456
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ ./ass1.pl 12 14 18 56 78 63 88 99 100 122
The Number List :12 14 18 56 78 63 88 99 100 122
Finding Average of the Numbers:
The Average Number is : 65
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ 
```

**Figure 1.2** In the screenshot the user provides input as command line arguments and the terminal window displays the average of number.

## Inference:

1. Writing PERL Script and how to execute it in Terminal window.
2. Get familiarised with Scalar Data, Arrays and List Data, Control Structure, Hashes syntax and using in the script.
3. Learning how to write a subroutine and calling the subroutine.

**Date:18/10/2021**

## Question 2:

Find and fix the bugs in the following program. Execute the correct program and show the output.

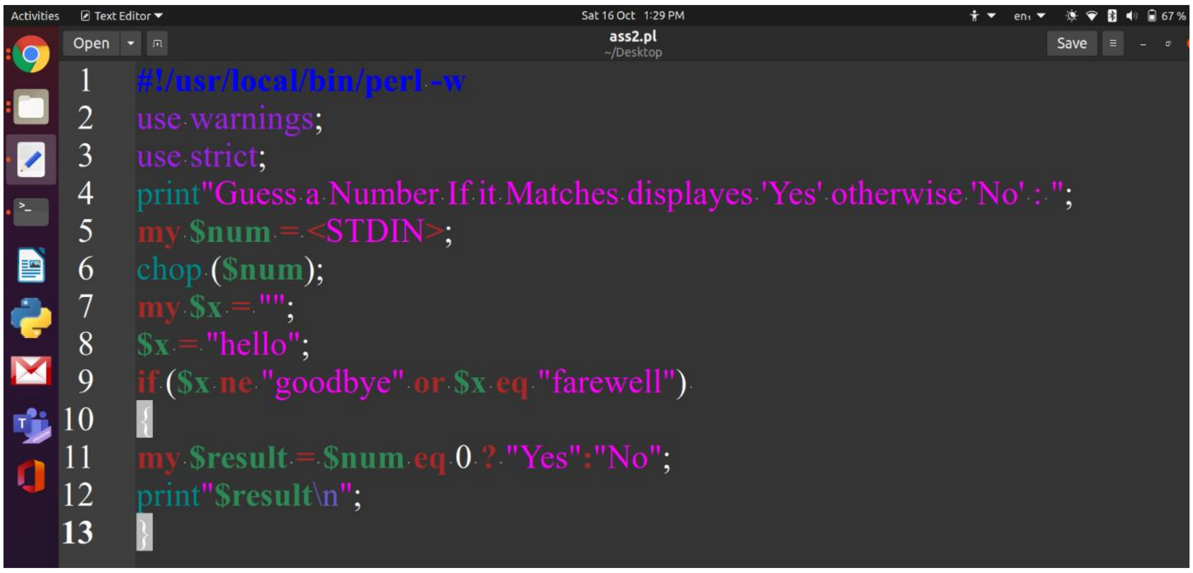
```
#!/usr/local/bin/perl
$num = <STDIN>;
chop ($num);
$x = "";
$x += "hello";
if ($x != "goodbye" | $x == "farewell") {
$result = $num eq 0 ? 43;
```

## Source Code after removing bugs:

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

use warnings;

use strict;

print"Guess a Number If it Matches displayes 'Yes' otherwise 'No' : ";

my $num = <STDIN>;

chop ($num); # It removes last character here it removes “\n”

my $x = ""; # Here my is private variable but since it is outside loop it is global

$x = "hello"; # Addition operation is not applicable for strings

if ($x ne "goodbye" or $x eq "farewell")

{

my $result = $num eq 0 ? "Yes":"No";

# Corrected Ternary Operator.

# If entered number equal to Zero prints yes else No.

print"$result\n";

}
```

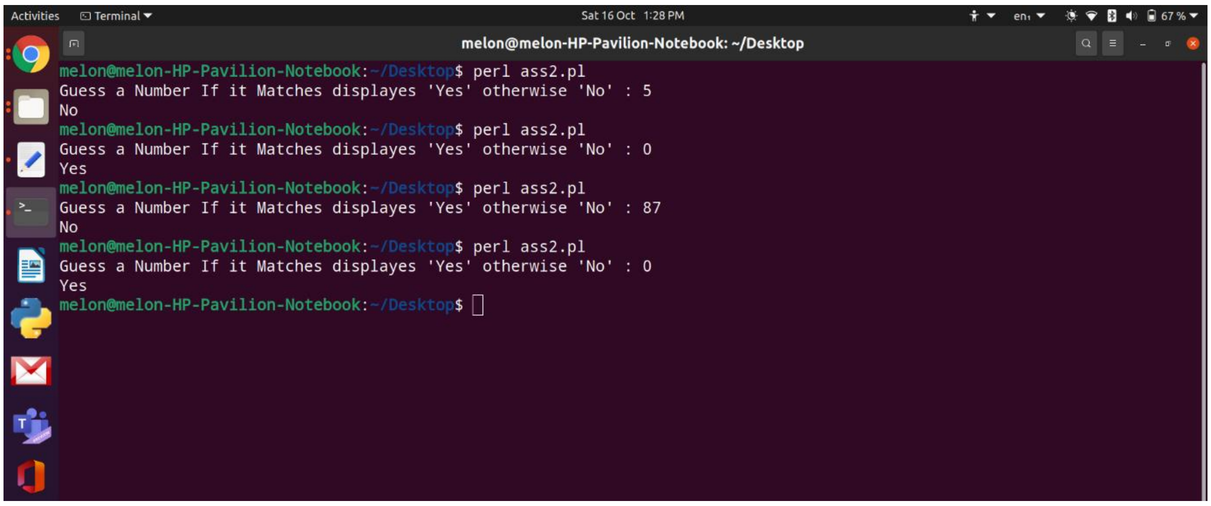
**Date:18/10/2021**

## PERL Code Screenshots:



```perl
#!/usr/local/bin/perl -w
use warnings;
use strict;
print"Guess a Number If it Matches displayes 'Yes' otherwise 'No' : ";
my $num = <STDIN>;
chop ($num);
my $x = "";
$x = "hello";
if ($x ne "goodbye" or $x eq "farewell")
{
my $result = $num eq 0 ? "Yes":"No";
print"$result\n";
}
```

**Figure 2.1** Screenshot of PERL code for removing the bugs.



```
melon@melon-HP-Pavilion-Notebook:~/Desktop$ perl ass2.pl
Guess a Number If it Matches displays 'Yes' otherwise 'No' : 5
No
melon@melon-HP-Pavilion-Notebook:~/Desktop$ perl ass2.pl
Guess a Number If it Matches displays 'Yes' otherwise 'No' : 0
Yes
melon@melon-HP-Pavilion-Notebook:~/Desktop$ perl ass2.pl
Guess a Number If it Matches displays 'Yes' otherwise 'No' : 87
No
melon@melon-HP-Pavilion-Notebook:~/Desktop$ perl ass2.pl
Guess a Number If it Matches displays 'Yes' otherwise 'No' : 0
Yes
melon@melon-HP-Pavilion-Notebook:~/Desktop$ 
```

**Figure 2.2** Screenshot of output of corrected PERL code.

## Inference:

1. Writing PERL Script and how to execute it in Terminal window.
2. Get familiarised with Scalar Data, Arrays and List Data, Control Structure, using in the script.
3. The arithmetic operation is not applicable to string data.

**Date:18/10/2021**

## Question 3 :

Write a program that prompts the user for a filename, then reads that file in and displays the contents backwards, line by line, and character-by-character on each line. You can do this with scalars, but an array is much easier to work with.
If the original file is: abcdef
ghijkl

the output will be: lkjihg

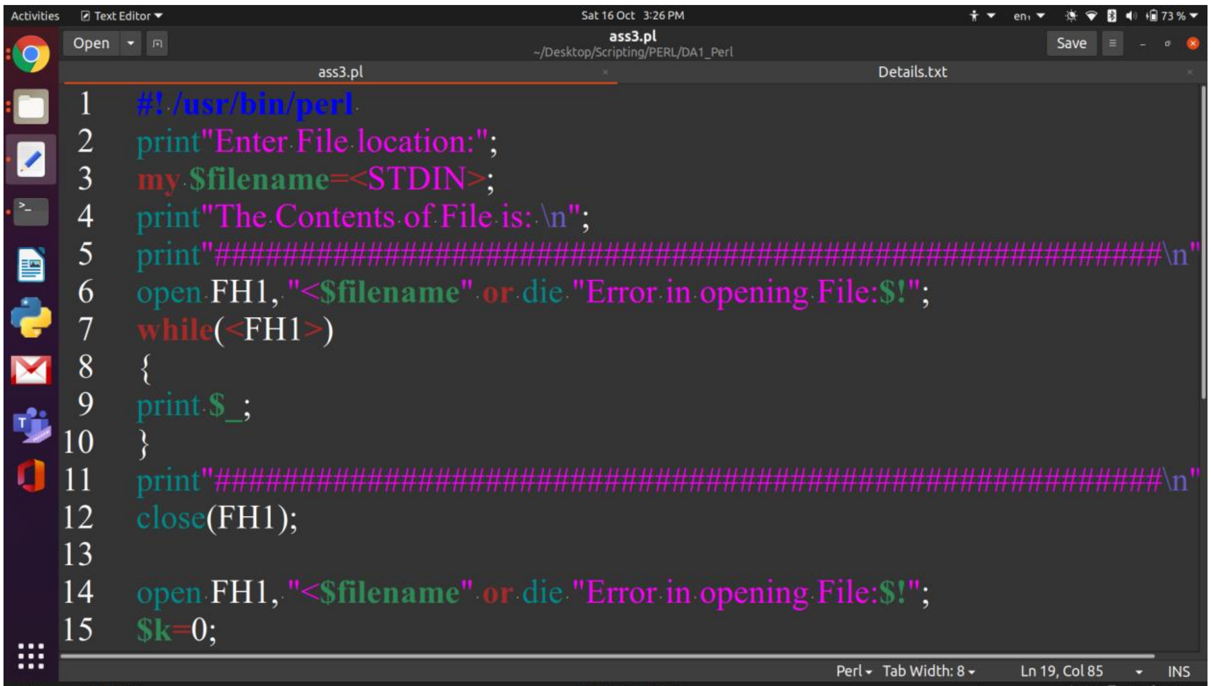
fedcba.

## Source Code:

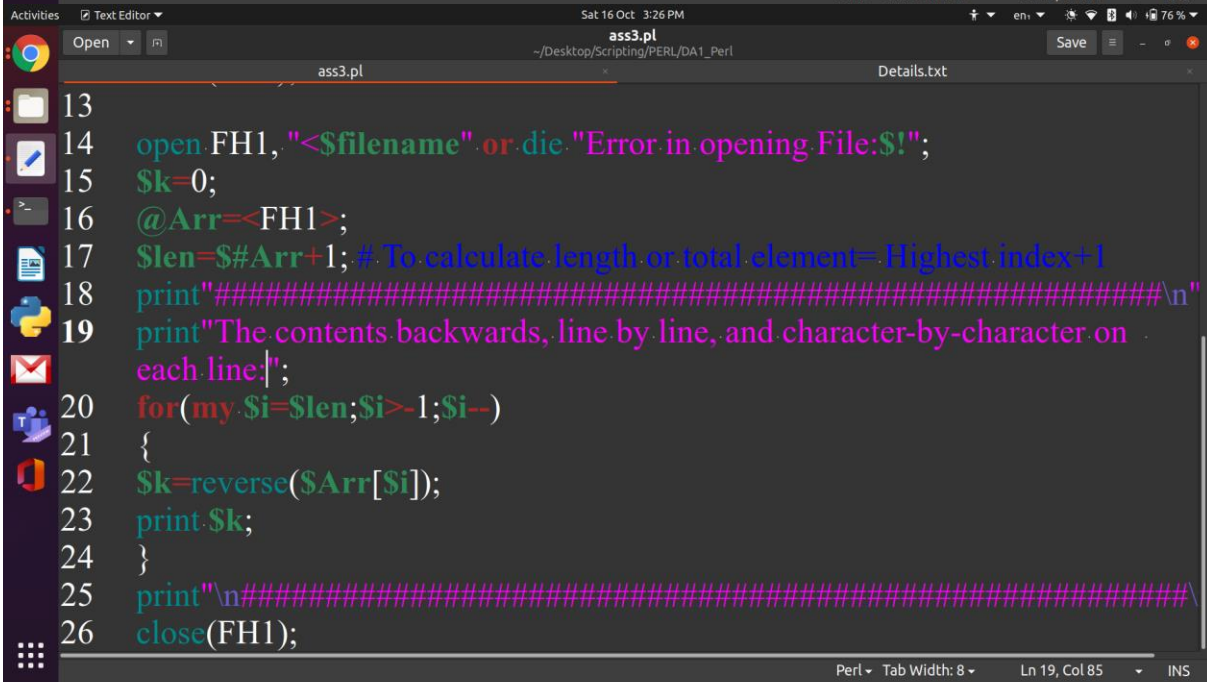
```
#! /usr/bin/perl

print"Enter File location:";

my $filename=<STDIN>;

print"The Contents of File is: \n";

print"########################################################\n";

open FH1, "<$filename" or die "Error in opening File:$!";

while(<FH1>)

{

print $_;

}

print"########################################################\n";

close(FH1);

open FH1, "<$filename" or die "Error in opening File:$!";

$k=0;

@Arr=<FH1>;

$len=$#Arr+1; # To calculate length or total element= Highest index+1

print"########################################################\n";

print"The contents backwards, line by line, and character-by-character on each line:";

for(my $i=$len;$i>-1;$i--)

{

$k=reverse($Arr[$i]);

print $k;

}
```

```
print"\n############################################################\n";
close(FH1);
```

## PERL Code Screenshots:



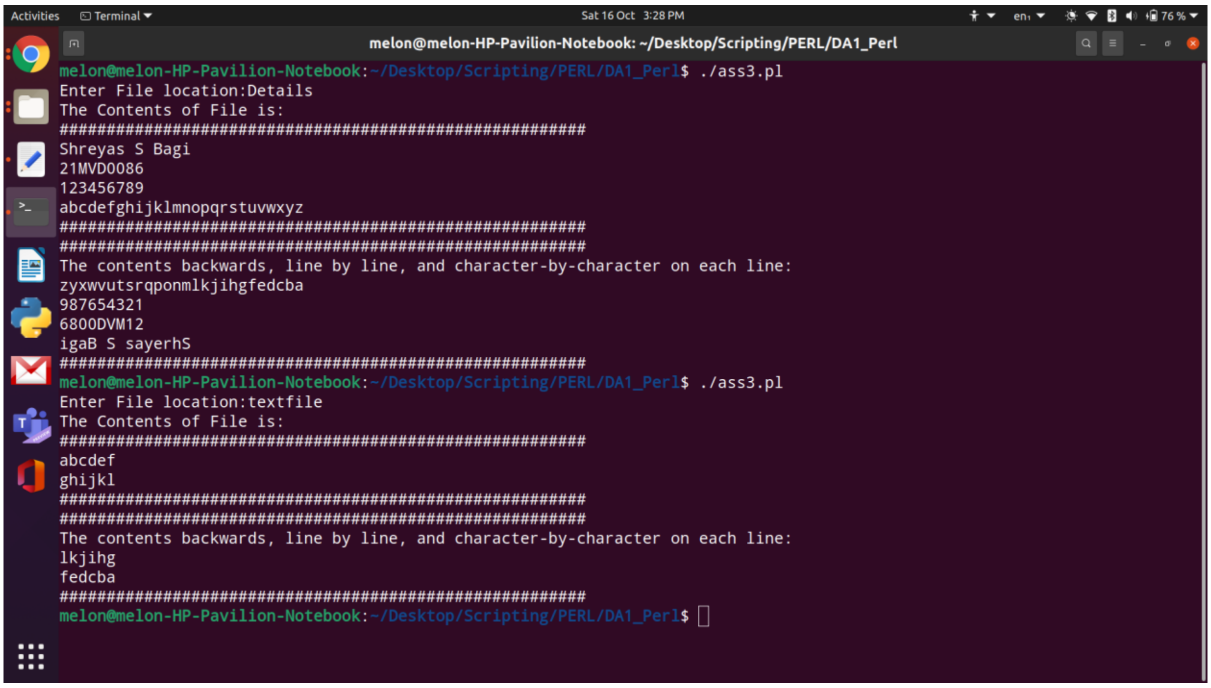
```
#! /usr/bin/perl
print"Enter File location:";
my $filename=<STDIN>;
print"The Contents of File is: \n";
print"################################################################\n"
open FH1, "<$filename" or die "Error in opening File:$!";
while(<FH1>)
{
print $_;
}
print"################################################################\n"
close(FH1);

open FH1, "<$filename" or die "Error in opening File:$!";
$k=0;
```



```

open FH1, "<$filename" or die "Error in opening File:$!";
$k=0;
@Arr=<FH1>;
$len=$#Arr+1; # To calculate length or total element= Highest index+1
print"################################################################\n"
print"The contents backwards, line by line, and character-by-character on 
each line:";
for(my $i=$len;$i>-1;$i--)
{
$k=reverse($Arr[$i]);
print $k;
}
print"\n################################################################\
close(FH1);
```

**Figure 3.1** The Screenshot shows us the PERL script for reversing the line and reversing the text written.

## Output Screenshot:


Activities
Terminal
Sat 16 Oct 3:28 PM
melon@melon-HP-Pavilion-Notebook: ~/Desktop/Scripting/PERL/DA1_Perl

```
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ ./ass3.pl
Enter File location:Details
The Contents of File is:
########################################################
Shreyas S Bagi
21MVD0086
123456789
abcdefghijklmnopqrstuvwxyz
########################################################
########################################################
The contents backwards, line by line, and character-by-character on each line:
zyxwvutsrqponmlkjihgfedcba
987654321
6800DVM12
igaB S sayerhS
########################################################
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ ./ass3.pl
Enter File location:textfile
The Contents of File is:
########################################################
abcdef
ghijkl
########################################################
########################################################
The contents backwards, line by line, and character-by-character on each line:
lkjihg
fedcba
########################################################
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$
```

**Figure 3.2** The screenshot reads the content of file that is present in current directory and output is also displayed.

## Inference:

1. Writing PERL Script and how to execute it in Terminal window.
2. Get familiarised with Scalar Data, Arrays and List Data, Control Structure, and using in the script.
3. Getting familiarised with Loops and inbuilt functions.
4. File handling, File manipulation and modes of file.

**Date:18/10/2021**

## Question 4 :

Create an array that holds the numbers from 1 to 10. Create a slice of that list with a different name that holds all the odd numbers, and another slice of that sublist that holds all the primes. Write a program that displays the list elements in each of the three lists. Also, display the size of all three lists.

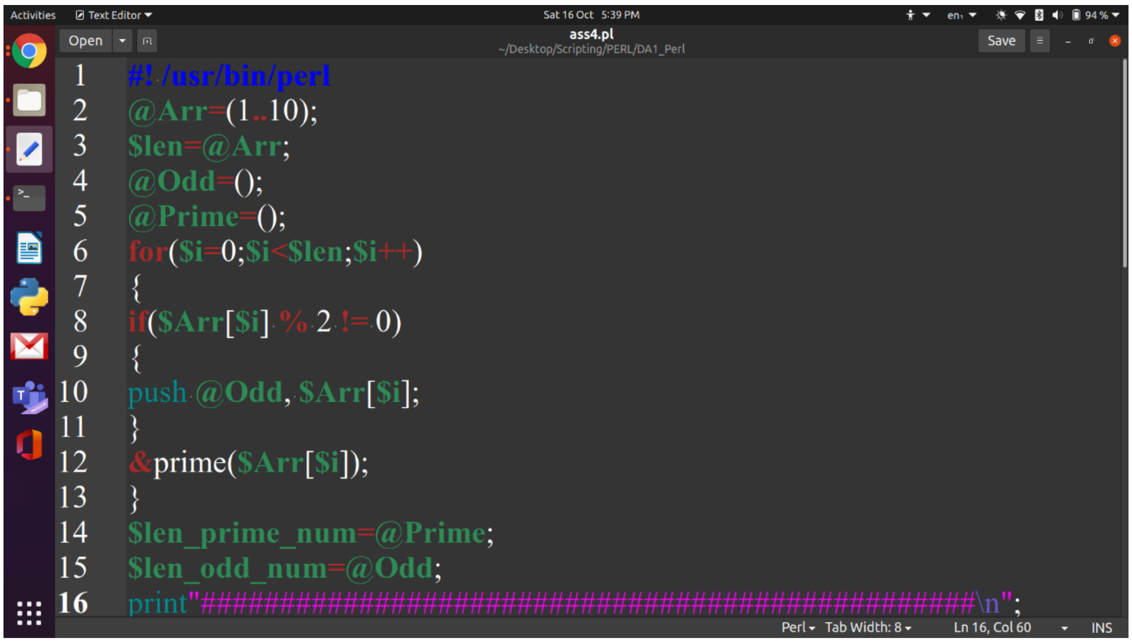
## Source Code:

```
#! /usr/bin/perl

@Arr=(1..10);

$len=@Arr;

@Odd=();

@Prime=();

for($i=0;$i<$len;$i++)

{

if($Arr[$i] % 2 != 0)

{

push @Odd, $Arr[$i];

}

&prime($Arr[$i]);

}

$len_prime_num=@Prime;

$len_odd_num=@Odd;

print"#####################################################\n";

print"The Array List:@Arr\n";

print"The length of Array List: $len\n";

print"#####################################################\n";

print"Print the Odd Array:@Odd\n";

print"The length of Odd Array List: $len_odd_num\n";

print"#####################################################\n";

print"Print the Prime Numbers:@Prime\n";

print"The length of Prime Number Array List: $len_prime_num\n";

print"#####################################################\n";
```

**Date:18/10/2021**

```
sub prime
{
$num=$_[0];
$count=0;
for($k =1; $k<=$num; $k++)
{
if($num % $k == 0)
{
$count +=1;
}
}
if ($count == 2)
{
push @Prime, $num;
}
}
```

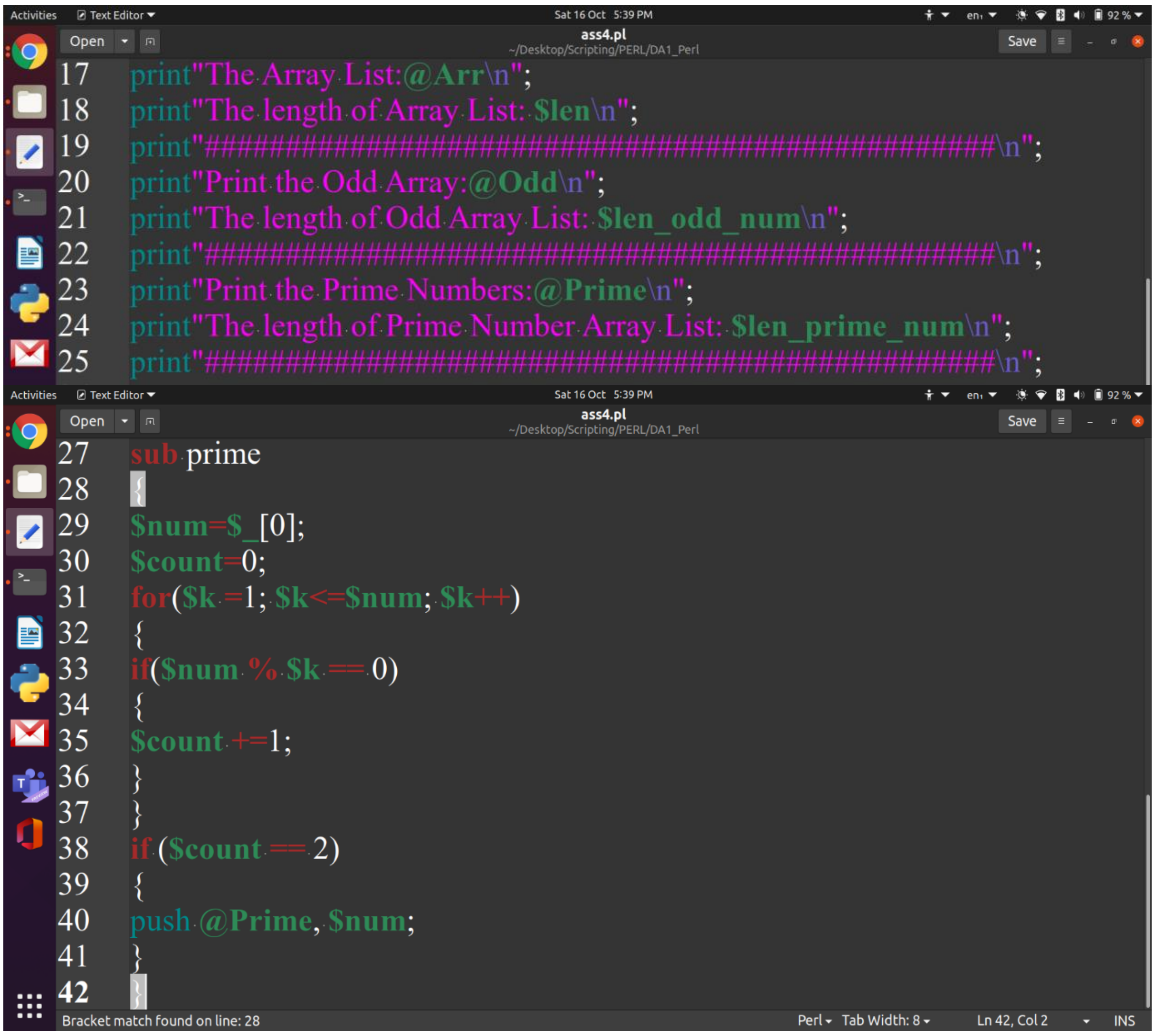
## PERL Code Screenshots :



```
#!/usr/bin/perl
@Arr=(1..10);
$len=@Arr;
@Odd=();
@Prime=();
for($i=0;$i<$len;$i++)
{
if($Arr[$i] % 2 != 0)
{
push @Odd, $Arr[$i];
}
&prime($Arr[$i]);
}
$len_prime_num=@Prime;
$len_odd_num=@Odd;
print"#################################################\n";
```

**Figure 4.1** The screenshot shows the PERL script written for Odd number and Prime number.

**Date:18/10/2021**



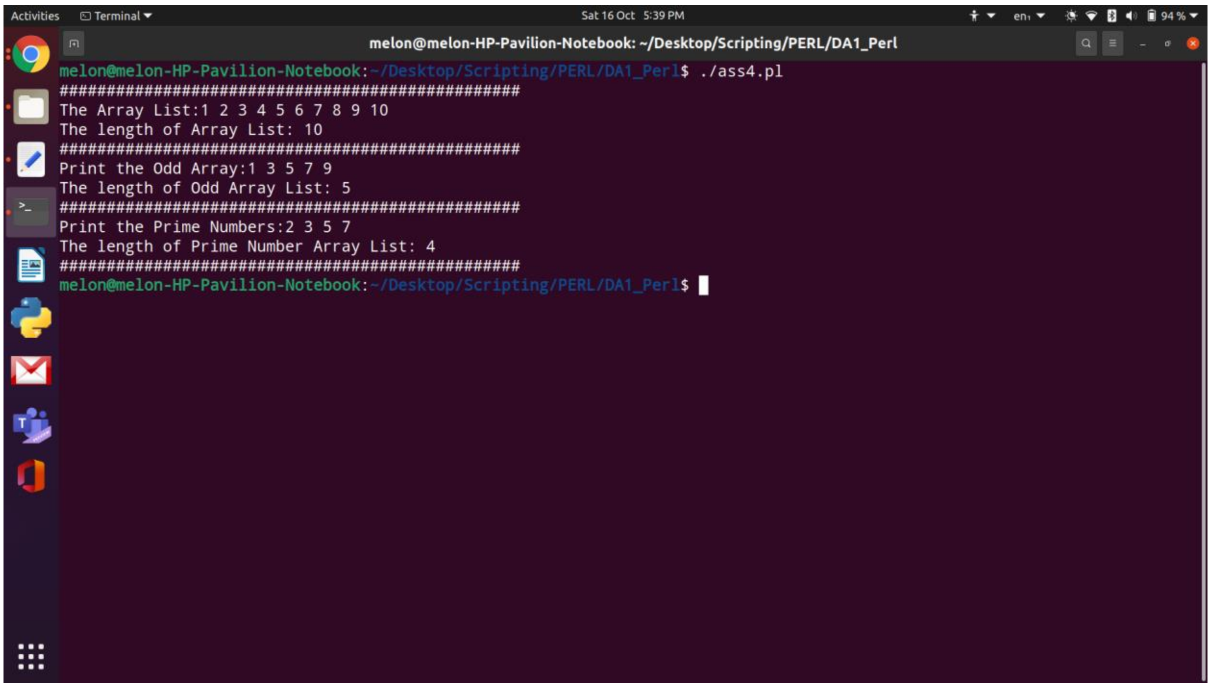
```perl
print"The Array List:@Arr\n";
print"The length of Array List: $len\n";
print"#####################################################\n";
print"Print the Odd Array:@Odd\n";
print"The length of Odd Array List: $len_odd_num\n";
print"#####################################################\n";
print"Print the Prime Numbers:@Prime\n";
print"The length of Prime Number Array List: $len_prime_num\n";
print"#####################################################\n";
```

```perl
sub prime
{
$num=$_[0];
$count=0;
for($k =1; $k<=$num; $k++)
{
if($num % $k == 0)
{
$count +=1;
}
}
if ($count == 2)
{
push @Prime, $num;
}
}
```

**Figure 4.2** The screenshot shows the PERL script written for Odd number and Prime number

Date:18/10/2021

## Output Screenshot:



```
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$ ./ass4.pl
#################################################
The Array List:1 2 3 4 5 6 7 8 9 10
The length of Array List: 10
#################################################
Print the Odd Array:1 3 5 7 9
The length of Odd Array List: 5
#################################################
Print the Prime Numbers:2 3 5 7
The length of Prime Number Array List: 4
#################################################
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/DA1_Perl$
```

**Figure 4.3** The screenshot segregates the input as two array one with Odd number and other with Prime number.

## Inference:

1. Writing PERL Script and how to execute it in Terminal window.
2. Get familiarised with Scalar Data, Arrays and List Data, Control Structure, and using in the script.
3. Getting familiarised with Loops and inbuilt functions.
4. File handling, File manipulation and modes of file.
5. Learning how to write a subroutine and calling the subroutine.