

### Experiment No- 10

**Aim: Execute the following awk/perl scripts:**

- (i) Write an awk script to print all even numbers in a given range.**
- (ii) Write an awk script to develop a Fibonacci series (take user input for number of terms).**
- (iii) Write a perl script to sort elements of an array.**
- (iv) Write a perl script to check if a number is prime or not.**

Roll No.	17
Name	Manav Jawrani
Class	D10A
Subject	Unix Lab
Lab Outcome	LO4: To Execute the following scripts using grep / sed commands
Date of Performance/ Submission	28/4/2022 - 30/04/2022

**Aim:** Execute the following scripts using awk / perl languages:

- (i) Write an awk script to print all even numbers in a given range.
- (ii) Write an awk script to develop a Fibonacci series (take user input for number of terms).
- (iii) Write a perl script to sort elements of an array.
- (iv) Write a perl script to check if a number is prime or not.

**Theory:**

**What is AWK?**

Awk is a scripting language used for manipulating data and generating reports. The awk command programming language requires no compiling and allows the user to use variables, numeric functions, string functions, and logical operators.

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that match with the specified patterns and then perform the associated actions. Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan.

**1. AWK Operations:**

- (a) Scans a file line by line
- (b) Splits each input line into fields
- (c) Compares input line/fields to pattern
- (d) Performs action(s) on matched lines

**2. Useful For:**

- (a) Transform data files
- (b) Produce formatted reports

**3. Programming Constructs:**

- (a) Format output lines
- (b) Arithmetic and string operations
- (c) Conditionals and loops

## What is Perl?

Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

- Perl is a stable, cross platform programming language.
- Though Perl is not officially an acronym, few people used it as Practical Extraction and Report Language.
- It is used for mission critical projects in the public and private sectors.
- Perl is an *Open Source* software, licensed under its *Artistic License*, or the *GNU General Public License (GPL)*.
- Perl was created by Larry Wall.
- Perl 1.0 was released to usenet's alt.comp.sources in 1987.
- At the time of writing this tutorial, the latest version of perl was 5.16.2.
- Perl is listed in the *Oxford English Dictionary*.

## Perl Features

- Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.
- Perl's database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.
- Perl works with HTML, XML, and other mark-up languages.
- Perl supports Unicode.
- Perl is Y2K compliant.
- Perl supports both procedural and object-oriented programming.
- Perl interfaces with external C/C++ libraries through XS or SWIG.
- Perl is extensible. There are over 20,000 third party modules available from the Comprehensive Perl Archive Network (CPAN).
- The Perl interpreter can be embedded into other systems.

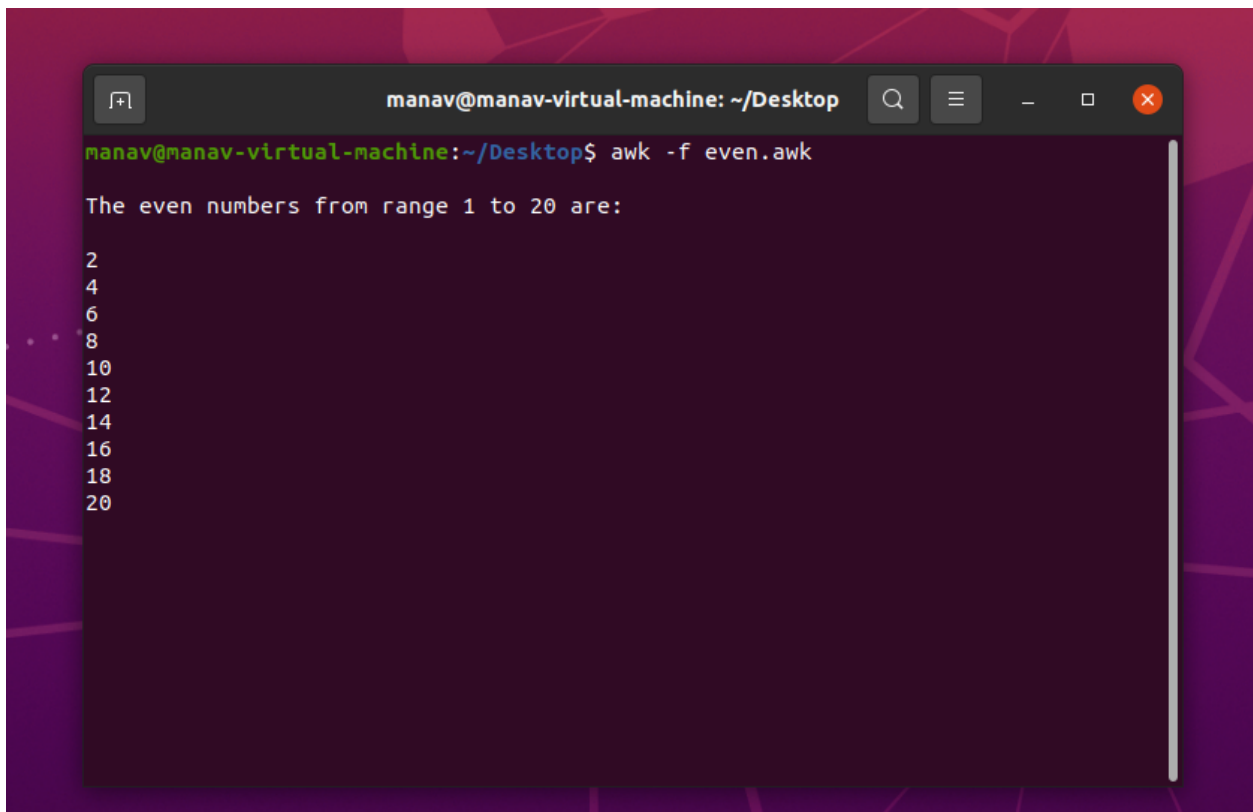
## **Implementation:**

### **A. AWK script to print all even numbers in a given range.**

#### **Code:**

```
$ awk  
{  
print "The even numbers from range 1 to 20 are:\n"  
{ for (i = 1; i <= 20; ++i) { if (i % 2 == 0) print i ; else continue} }  
}
```

#### **Output:**



```
manav@manav-virtual-machine: ~/Desktop  
manav@manav-virtual-machine:~/Desktop$ awk -f even.awk  
The even numbers from range 1 to 20 are:  
2  
4  
6  
8  
10  
12  
14  
16  
18  
20
```

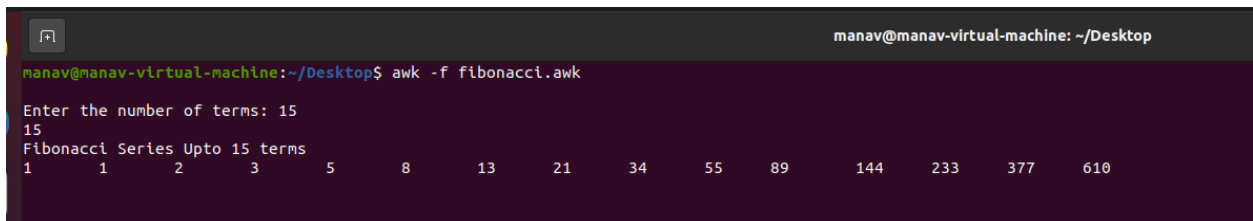
## B. AWK script to develop a Fibonacci series (take user input for number of terms).

### Code:

```
{
printf "Enter the number of terms: "
getline terms < "-"
printf ("%s\n",terms)
fib[1] = 1
fib[2] = 1
for (i=3; i<=terms; i++)
fib[i] = fib[i-1]+fib[i-2]

printf("Fibonacci Series Upto %d terms\n",terms)
for(i=1;i<=terms;i++)
printf("%d\t",fib[i])
}
```

### Output:



The screenshot shows a terminal window with the following content:

```
manav@manav-virtual-machine: ~/Desktop
manav@manav-virtual-machine:~/Desktop$ awk -f fibonacci.awk
Enter the number of terms: 15
15
Fibonacci Series Upto 15 terms
1      1      2      3      5      8      13     21     34     55     89     144    233    377    610
```

The output displays the first 15 terms of the Fibonacci series, with each term separated by a tab character.

### C. Perl script to sort elements of an array.

#### Code:

```
#!/usr/bin/perl

# Initializing an array
@numbers = (53, 17, 27, 3, 51);

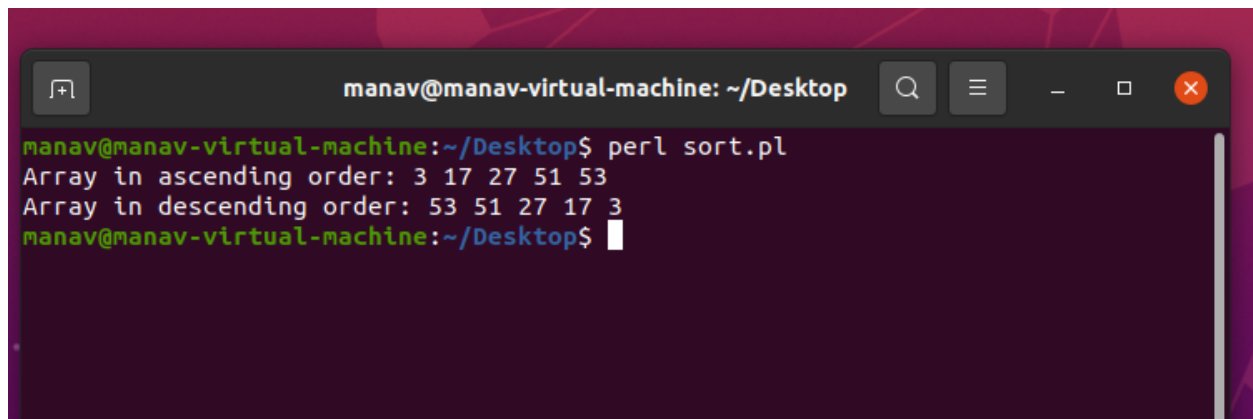
# Sorting array in ascending order
@x = sort { $a <=> $b } @numbers;

# Sorting array in descending order
@y = sort { $b <=> $a } @numbers;

# Printing sorted array
print "Array in ascending order: @x\n";

# Printing sorted array
print "Array in descending order: @y\n";
```

#### Output:

A terminal window titled "manav@manav-virtual-machine: ~/Desktop" with standard window controls. The terminal shows the execution of a Perl script named "sort.pl". The output of the script is displayed in two lines: "Array in ascending order: 3 17 27 51 53" and "Array in descending order: 53 51 27 17 3". The prompt "manav@manav-virtual-machine:~/Desktop\$" is visible at the end of the second line of output.

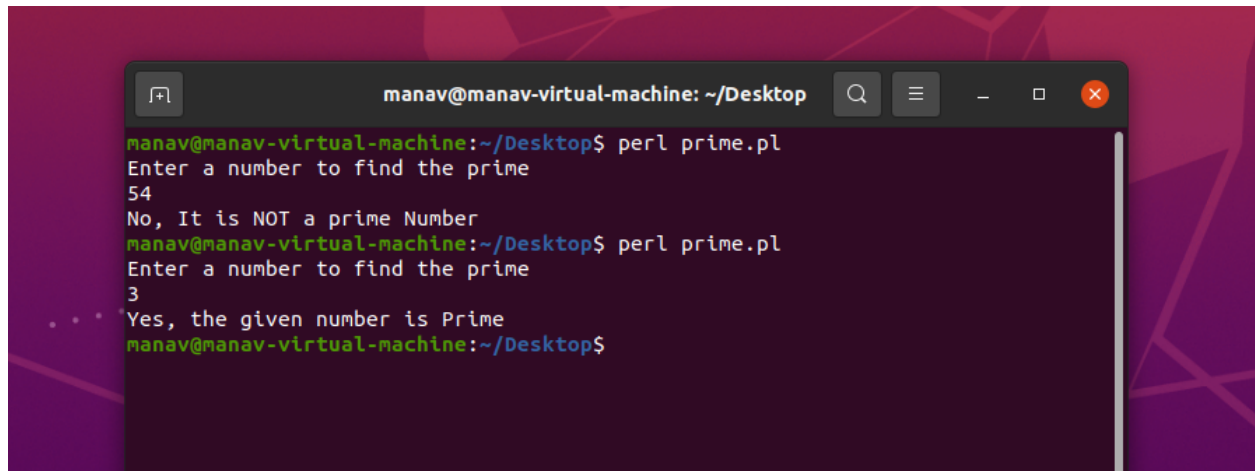
```
manav@manav-virtual-machine:~/Desktop$ perl sort.pl
Array in ascending order: 3 17 27 51 53
Array in descending order: 53 51 27 17 3
manav@manav-virtual-machine:~/Desktop$
```

**D. Perl script to check if a number is prime or not.**

**Code:**

```
#!/usr/bin/perl
sub testprime
{
my $m = shift @_;
my $i = 2;
while ($i < $m)
{
return 0 unless ($m % $i++);
}
return 1;
}
print "Enter a number to find the prime \n";
chomp (my $n = <STDIN>);
my $FindPrime = testprime $n;
if ( $FindPrime == 1)
{
print "Yes, the given number is Prime \n";
}
else
{
print "No, It is NOT a prime Number \n";
}
```

## Output:

A terminal window titled 'manav@manav-virtual-machine: ~/Desktop' with standard window controls. The terminal shows the execution of a Perl script 'perl prime.pl'. The first run takes input '54' and outputs 'No, It is NOT a prime Number'. The second run takes input '3' and outputs 'Yes, the given number is Prime'.

```
manav@manav-virtual-machine:~/Desktop$ perl prime.pl
Enter a number to find the prime
54
No, It is NOT a prime Number
manav@manav-virtual-machine:~/Desktop$ perl prime.pl
Enter a number to find the prime
3
Yes, the given number is Prime
manav@manav-virtual-machine:~/Desktop$
```

## Conclusion:

Thus we have understood awk and perl scripting and executed the scripts.