ASSIGNMENT NO. 01

PROGRAM STATEMENT:

Write a shell script/program to generate Fibonacci Series up to a certain number.

THEORY:

In mathematics, the **Fibonacci numbers** or **Fibonacci sequence** are the numbers in the following integer sequence
1,1,2,3,5,8,13,21,34,55,89,144.........
or (often, in modern usage):
0,1,1,2,3,5,8,13,21,34,55,89,144.........

By definition, the first two numbers in the Fibonacci sequence are either 1 and 1, or 0 and 1, depending on the chosen starting point of the sequence, and each subsequent number is the sum of the previous two.
In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the recurrence relation
$F_n=F_{n-1}+F_{n-2}$
with seed values
$F_1=1,F_2=1$
or
$F_0=0,F_1=1$
The Fibonacci sequence is named after *Fibonacci.* His 1202 book *Liber Abaci* introduced the sequence to Western European mathematics,although the sequence had been described earlier in Indian mathematics. By modern convention, the sequence begins either with $F_0 = 0$ or with $F_1 = 1$. The *Liber Abaci* began the sequence with $F_1 = 1$.

ALGORITHM:

Steps:

```
1. Ask user for the number of inputs.
2. Initialize first two numbers x and y by 0 and 1 respectively.
3. Initialize i by 2.
4. Print the values of x and y.
5. while(i<n)
   do
   i)      Increment the value of i.
   ii)     z=x+y(take a third variable z to store the sum of first two variables a
           and b)
   iii)    Print the values of x y and z.
6. End while loop.
```

PROGRAM SCRIPT (CODE):

```
echo "How many number of terms to be generated ?"
read n
x=0
y=1
i=2
echo "Fibonacci Series up to $n terms :"
echo "$x"
echo "$y"
while [ $i -lt $n ]
do
i=`expr $i + 1 `
z=`expr $x + $y `
echo "$z"
x=$y
y=$z
done
```

INPUT:

```
bash fibonacci.sh
```

OUPUT:

```
How many number of terms to be generated ?
10
Fibonacci Series up to 10 terms :
0
1
1
2
3
5
8
13
21
34
```

DISCUSSIONS:

i) Fibonacci series for a given number could also be generated based on this program.

ii) This program is based on addition thus having less mathematical computations.

iii) This program requires an array which stores all the numbers in that series.This ensures retrival of any ith term(i<n) of the series.

iv) This program could be approached in an recursive way.

<u>ASSIGNMENT NO. 02</u>

<u>PROGRAM STATEMENT:</u>
Write a shell script/program to print a reverse number of a given number.

<u>ALGORITHM:</u>
<u>Steps:</u>

```
Ask user for a number.
Read n.
Check for positive by checking its sign.
Store the number of characters(digits) in the number,l.
Set i:=1
Initialize a string t with “”.
for i<=l
do
        c=use “cut” command to cut the ith digit(character) from the number
         t=ct(string concatenation)
         i=i+1
end of step 7 for
Display t as the reverse number.
End.
```

PROGRAM SCRIPT(CODE):

```
clear
c=0
i=1
t=""
while [ $c -lt 2 ]
do
echo "Enter Your Positive Number"
read n
d=` echo $n|cut -c 1 `
if [ "$d" = "-" ]
then
echo "Wrong Entry,Try once"
let c=$c+1
elif [ $c -eq 2 ]
then
exit
```

```
else
echo "Number Given:"$n
f=` echo $n|wc -c `
while [ $i -ne $f ]
do
b=` echo $n|cut -c $i `
t=$b$t
let i=$i+1
done
echo  "Reverse:"$t
let c=$c+3
fi
done
```

INPUT:

```
$sh reverse.sh #(original script file)
```

OUPUT:

Enter Your Positive Number
-2897(*Input*)
Wrong Entry,Try once(*Output*)
7856(*Input*)
Number Given:7856
Reverse:6587(*Output*)

DISCUSSIONS:

i)This number is basically based on string concatenation;Unix takes any input as string.
ii) This program can be applied for checking palindrome number.
iii) This program can be further developed.

ASSIGNMENT NO. 03

PROGRAM STATEMENT:
Write a shell/program that reads an integer and test whether it is divisible by 11 using divisibility rule.

THEORY:
A number passes the test for 11 if the difference of the sums of alternating digits is divisible by 11.(This abstract and confusing sounding rule is much clearer with a few examples).
946 → (9+6) - 4 = 11 which is, of course, evenly divided by 11 so 946 passes this divisibility test
10,813 → (1+8+3) - (0+1) = 12-1 =11. Yes, this satisfies the rule for 11!
25, 784 = → (2+ 7 + 4) - (5+8) = 13 - 13 =0 . Yes, this does indeed work. In case you found this last bit confusing, remember that any number evenly divides 0. Think about it, how many 11's are there in 0? None, right. Well that means that 11 divides zero, zero times!
119,777,658 → (1+ 9 + 7 + 6 + 8) - (1+ 7 + 7 +5) = 31 - 20 = 11

ALGORITHM:

Steps:
Read the number from user,n
Cut the numbers from odd positions of that number and store their some in a variable,o,using "**cut**" command.
Cut the numbers from even positions of that number and store their some in a variable,e,using "**cu**t" command.
If the difference between o and e i.e., o-e=0/multiple of 11,then the number is divisible by 11,else not.
End.

PROGRAM CODE(SCRIPT):

```
    echo  "Please enter the number";read n
o=0
e=0
l=` echo $n|wc -c `
let l=$l-1
for (( i=1 ; i<=l ; i++ ))
do
r=` expr $i \% 2 `
u=` echo $n|cut -c $i `
if [ $r -eq 0 ]
then
e=`expr $e + $u`
else
let o=$o+$u
fi
done
o=`expr $o - $e`
o=`expr $o \% 11`
if [ $o -eq  0 ]
then
echo "Its divisible"
else
echo "Not divisible"
fi
```

INPUT:

```
sh testing11div.sh
```

OUTPUT:

```
Please enter the number
121
Its divisible
```

INPUT:

```
sh testing11div.sh
```

OUTPUT:

```
Please enter the number
345
```

```
Not divisible
```

DISCUSSION:

This program shows the logic of divisibility of a number with 11.
This program can be modified for dividing with 2,3,etc.
This program is much difficult approach.A simple one involves much simple logic.

ASSIGNMENT NO.04

PROGRAM STATEMENT:

Write a program to print the following pattern:

i)

```
*
**
***
****
*****
```

Theory: This program is nothing but to print the above mentioned pattern using shell scripting and to display the output properly.Here the no of rows will be given dynamically.

Algorithm:

```
Step1:print "Please enter the number of lines"
read n
Step2:i<-1
Step3:Repeat step 4 to step 8 till i<=n
Step4:j<-1
Step5:Repeat step 6 to step 7 till j<=i
Step6:print "*"
Step7:j<-j+1
[end of step5 loop]
Step8: i<-i+1
[end of step 3 loop]
Step9: print "\n"
Step 10: Stop
```

Source Code:

```
read -p "Please enter the number of lines:"
for (( i=1 ; i<=$n ; i++ ))
do
        for(( j=1 ; j<=$i ; j++ ))
        do
        echo -ne "*"
        done
echo -e "\n"
done
```

Output:

```
Please enter the number of lines:5
*
**
***
****
*****
```

PROGRAM STATEMENT:

ii)

```
                *
            *   *   *
        *   *   *   *   *
    *   *   *   *   *   *   *
*   *   *   *   *   *   *   *   *
```

Theory: The objective of the progam is to print the pattern.Here we have taken the no. of rows from the user.

Algorithm:

```
Step 1:print "Enter the no. of rows"
read n
Step 2: i<-1
Step 3: Repeat step 4 to step 17 till i<=n
    Step 4: j<-i
    Step 5: Repeat step 6 to step 7 till j<=n
        Step 6: print " "
        Step 7:j<-j+1
    [end of step 5 loop]
    Step 8: k<-1
    Step 9: Repeat step 10 to step 11 till k<=i
        Step 10:  print "*"
        Step 11: k<-k+1
    [end of step 9 loop]
    Step 12:z<-1
    Step 13:Repeat step 14 to step 15 till z<i
        Step 14: print "*"
        Step 15: z<-z+1
    [end of step 13 loop]
```

Step 16: print "\n"

Step 17: i<-i+1

[end of step 3 loop]

Step 18: Stop

<u>Source Code:</u>

```
echo "Enter the no of lines"
read n
for(( i=1 ; i<=$n ; i++ ))
do
        for(( j=i ; j<=$n ; j++ ))
        do
        echo -ne "*"
        done
        for(( k=1 ; k<=$i ; k++ ))
        do
        echo -ne "*"
        done
        for(( z=1 ; z<$i ; z++ ))
        do
        echo -ne "*"
        done
echo -e "\n"
done
```

Output:

```
Enter the no of lines

5

                    *

                *   *   *

            *   *   *   *   *

        *   *   *   *   *   *   *

    *   *   *   *   *   *   *   *   *
```

PROGRAM STATEMENT:

iii)

```
1
0   1
0   1   0
1   0   1   0
1   0   1   0   1
```

Theory: We print the pattern of 1s and 0s.Here we take the no of rows from the user.

Algorithm:

```
Step 1: print"Enter the value of n to continue"
read n
Step 2: print "\n"
Step 3: i<-0
j<-0
k<-1
l<-0
Step 4: Repeat step 5 to step 12 till i<n
    Step 5: Repeat step 6 to step 9 till j<i
        Step 6: print k
        Step 7: j<-j+1
        Step 8: l<-k+1
        Step 9: k<-l%2
    [end of step 5 loop]
    Step 10: print "\n"
    Step 11: j<-0
    Step 12: i<-i+1
[end of step 4 loop]
```

Step 13: Stop

<u>Output:</u>

```
Enter the value of n to continue

5

1

0       1

0       1       0

1       0       1       0

1       0       1       0       1
```

iv)Print the value of sinx.

$\sin(x)=x-x^3/3!+x^5/5!-x^7/7!+.....$

Theory: Here we have printed the sine series.The value of x and no of terms are taken from the user.

Algorithm:

```
Step 1: print"Enter no of terms"
read n
Step 2: print "Enter the value of x"
read x
Step 3: x1<-x
Step 4: x<-(print x*3.14/180)
Step 5: t<-0
sum<-0
f<-1
j<- -1
Step 6: i<-1
Step 7:Repeat step 8 to step 18 till i<=n
Step 8:j<-j+2
f<-1
Step 9: p<-1
Step 10: Repeat step 11 to step 12 till p<=j
Step 11: f<-(print f*p)
Step 12: p<-p+1
[end of step 10 loop]
Step 13:if (i%2)!=0
then
Step 14: t<-(print x^j)
Step 15: sum<- (print sum+(t/f))
```

```
else
Step 16: t<-(print x^j)
Step 17:sum<-(print sum-(t/f))
[end of if]
Step 18: i<-i+1
[end of step 7 loop]
Step 19:print"sin(x1):sum"
Step 20:stop
```

Source Code:

```
echo -n "Enter no of terms"
read n
echo -n "Enter x: "
read x
x1=$x
x=$(echo "$x * 3.14/180" | bc -l)
t=0
sum=0
f=1
j=-1
for(( i=1 ; i<=n ; i++ ))
do
j=$(( $j + 2))
f=1
for(( p=1 ; p<=j ; p++ ))
do
f=$(echo "$f*$p" | bc -l)
done
if[ `expr $i % 2 -ne 0 ]
```

```
then
t=$(echo "$x ^ $j" | bc -l)
sum=$(echo "$sum + $t/$f" | bc -l)
else
t=$(echo "$x ^ $j" | bc -l)
sum=$(echo "$sum - $t / $f" | bc -l)
fi
done
echo "sin($x1):$sum
```

Output:

```
Enter No. of terms : 4
Enter x : 2
sin(2) : .03488181132605679132
```

Discussion: In this entire program we have printed different types of pattern by using the basic syntax of shell scripting.We have taken the number of rows dynamically from the user to make the programs more acceptable.

ASSIGNMENT NO. 05

PROGRAM STATEMENT:

Write a shell script/program to generate a possible combinations of 1, 2 and 3.

ALGORITHM:

Steps:

```
Take three variables i,j,k.
for i in 1 2 3
do
for j in 1 2 3
do
for k in 1 2 3
do
Display value of i,j,k
 End of step 6 loop.
  End of step 4 loop.
  End of step 2 loop.
  End.
```

PROGRAM CODE(SCRIPT):

```
clear
for in 1 2 3
do
for j in 1 2 3
do
for k in 1 2 3
do
echo $i  $j  $k
done
done
done
```

INPUT:

```
$ sh comb1.sh
```

OUTPUT:

```
#1 1 1
#1 1 2
```

```
#1 1 3
#1 2 1
#1 2 2
#1 2 3
#1 3 1
#1 3 2
#1 3 3
#2 1 1
#2 1 2
#2 1 3
#2 2 1
#2 2 2
#2 2 3
#2 3 1
#2 3 2
#2 3 3
#3 1 1
#3 1 2
#3 1 3
#3 2 1
```

DISCUSSION:

This is a simple combination based on positions of individual 1,2or 3.
This program could be developed further.
Based on other conditions various combinations could be made.

ASSIGNMENT NO. 06

PROGRAM STATEMENT:
Write a shell script/program to count number of lines and words in a file.

Algorithm:
Steps:
The file name is passed as command-line argument or read from the user.
The no. of lines and words present in the file is simply sent as an input to the "**wc**" command.
Display number of lines and words in that file after storing them in some variable.
End.

PROGRAM CODE:
```
clear
echo"Enter File Name"
read n
cat $n  #Showing contents
echo "Number of lines:"
echo ` wc –l $n `
echo "Number of words"
echo ` wc –w $n `
```

INPUT:
```
$sh countingfile.sh(original shell file)
```

OUTPUT:
```
Enter File Name
Test.txt
This file is just created for the testing of original file.
If it runs correctly then the program is running well.
We can see the number of lines and words in the file.
It should run quite well if rest is assured.
Number of lines:
4 Test.txt
Number of words:
42 Test.txt
```

Discussion:
This is basically the easiest way to achieve our target using “**wc**” command.
Other way is using “**exec**” command an setting a counter to count number of lines and words.
Further scope of developing programs lies there.

ASSIGNMENT NO. 07

PROGRAM STATEMENT:
Write a shell script/program to find the sum of N numbers taken as parameter input.

THEORY:
In mathematics, the **natural numbers** (sometimes called the **whole numbers**) are those used for counting (as in "there are *six* coins on the table") and ordering (as in "this is the *third* largest city in the country"). In common language, these purposes are distinguished by the use of cardinal and ordinal numbers, respectively. A third use of natural numbers is as nominal numbers, such as the model number of a product, where the "natural number" is used only for naming (as distinct from a serial number where the order properties of the natural numbers distinguish later uses from earlier uses) and generally lacks any meaning of *number* as used in mathematics but rather just shares the character set.
The natural numbers are a basis from which all other numbers may be built by extension: the integers, the rational numbers, the real numbers, the complex numbers, the hyperreal numbers, the nonstandard integers, and so on.[5][6]
Thereby the natural numbers are canonically embedded (identification) in the other number systems.
Thus sum of n numbers mean sum of first n natural numbers.

AlGORITHM:
Steps:

```
Read the number n from user.
Set n:=0
Set i:=1
for i<=n
do
n=n+i
i=i+1
End of step 3 loop.
Display the sum,n
End
```

PROGRAM CODE(SCRIPT):

```
clear
n=0
for (( i=1 ; i<=$1 ; i++ ))
do
n=` expr $i + $n `
done
echo "Thus the Sum upto"$1"th number from 1 is"
echo $n
```

INPUT:

```
$sh seriesn.sh 5
```

OUTPUT:

```
Thus the Sum upto5th number from 1 is
15
```

DISCUSSION:

Here basically a sum of n ***natural numbers*** is considered.
The series could generate sums for very large value of n.
This program could be modified a little to generate sum for an infinite series.
This program could be modified for accepting ***whole*** as well as ***integers*** for some global approach.

ASSIGNMENT NO. 08

PROGRAM STATEMENT:

A student examination files containing the following format:
Name sub1 sub 2 sub 3 sub 4
The students are awarded division as per following rules:
I) Percentage above or equal to 60- 1st division.
II) Percentage between 50 to 59- 2nd division.
III) Percentage between 40 to 49- 3rd division.
IV) Percentage less than 40- FAIL.
Write a shell script to the division awarded to each student.

THEORY:

This program is based on file programming.Reading from file is basically what we have to do.Here we have to compare data from a file.This includes complex ***if***structures.This works on *.txt* file.

ALGORITHM:

Steps:

Create a text file containing the details of the students as required.
Pass this file name as command-line argument into the main shell program.
Calculate the whole subjects marks and thus their percentage.
Based on this we have to calculate the division inside an if structure.
End.

PROGRAM SCRIPT(CODE):

```
clear
y=0
exec<$1
while read line
do
set $line
echo "Name of student:"$1 $2
j=3
for (( k=1 ; k<5 ; k++ ))
do
m=` echo $line|cut -d" " -f $j `
y=` expr $y \+ $m `
```

```
echo "Marks for subject"$k":"$m
let j=$j+1
done
echo $y
o=` echo "scale=2;($y/400)*100"|bc `
echo "Percentage of marks:"$o"%"
f=` echo $o|cut -d"." -f 1 `
if [ $f -gt 59 ]
then
echo "1st Division"
elif [ $f -gt 49 ]
then
echo "2nd Division"
elif [ $f -gt 39 ]
then
echo "3rd Division"
else
echo "Fail"
fi
done
```

INPUT:

```
$cat marks
Raja Sinha 71 85 60 74(original content)

$sh division.sh #(original script file)
```

OUTPUT:

```
Name of student:Raja Sinha
Marks for subject1:71
Marks for subject2:85
Marks for subject3:60
Marks for subject4:74
Percentage of marks:72.00%
1st Division(output)
```

DISCUSSION:
Basically a file program in unix.
This program could be developed for stating the division of a large number of students.
This could be also reffered to as a ***Database System program*** as it maintains and manipulates some data.
Further scope for operating on many such files could be made.

ASSIGNMENT NO. 09

PROGRAM STATEMENT:
Write a shell script which delete all lines contain the word "LINUX" from a given file.

THEORY:
This mainly includes the file searching operation using ***"grep"*** command."**grep**" command find a word or searches for a pattern from a file as its input.The concerned word could be present in plural,or as a single phrase.We should also consider the case of the case,i.e.,our program must resolve the case-sensitive dilemma.

ALGORITHM:
Steps:

Create a file in your system to avoid any unwanted modification.
Use "grep" command for finding those lines which were containing the word "Linux".
Write those lines in another file.
Replace the previous file with this newly created file permanently.
End.

PROGRAM SCRIPT(CODE):
```
clear
grep –v Linux test.txt>temp
mv temp test.txt
echo "Work Done"
```

INPUT:
```
$cat test.txt
Linux is an open-source Operating system.
Some Linux versions are Ubuntu,Fedora,etc.
Open-source softwares are free of cost.
Thus Linux like OSs could seize a good market hold from Windows.
Use of open-source softwares is increasing day by day.
Thus Linux has a good business oppurtunity.
It should remain free for atleast another 15 years for its rapid spread.
$sh dellinux.sh#(original script file)
```

OUTPUT:

```
Work Done
$cat test.txt
Open-source softwares are free of cost.
Use of open-source softwares is increasing day by day.
It should remain free for atleast another 15 years for its rapid spread.
```

DISCUSSION:

This involves deleting without actually deleting any content;just selecting some specific lines.
Since unix is an OS,we could tactically use the "*grep*" command here.
Program could betested with any other file;use of command line argument would be better.

ASSIGNMENT NO. 10

PROGRAM STATEMENT:

Write a shell script minimum of n numbers within an array element.

ALGORITHM:

Steps:

```
1. Ask user for the number of inputs.
2. for(i=0;i<n;i++)
   1) Take the elemements in array.
3. End for loop
4. Take a variable min where the first element of array is stored.
5. for(i=0;i<n;i++)
   1) if(min>a[i])
          then min=a[i]
      end if.
6. End for loop.
7. Print the minimum number.
```

PROGRAM SCRIPT(CODE):

```
echo -ne " Enter the number of elements >> "
read n
for((i=0;i<n;i++))
do
        echo -ne " Enter data >> "
        read a[$i]
done
min=${a[0]}
for ((i=0;i<n;i++))
do
        if [ $min -gt ${a[$i]} ]
        then
                min=${a[$i]}
        fi
done
echo "The minimum number is >> $min "
```

INPUT:

```
bash min.sh
```

OUPUT:

```
Enter the number of elements >> 7
 Enter data >> 3
 Enter data >> 7
 Enter data >> 2
 Enter data >> 1
 Enter data >> 18
 Enter data >> 8
 Enter data >> 9
The minimum number is >> 1
```

DISCUSSIONS:

1. Here we store the elements in an integer array.
2. In this program we are not using any function to calculate the min value but if we want to calculate this min value we can do this program by using recursive function.

ASSIGNMENT NO. 11

PROGRAM STATEMENT:
Write a shell script to check the types of the input file name.

Theory:

Here we have taken the file name from the user and used the cut command and cut the delimeter after the '.' and print the extension.

Algorithm:

```
Step 1: print"Enter the filename: "
read f
Step 2: if (-f in f) then
Step 3: [cut the file name after the delimeter '.']
ch<-(print f)
Step 4: print ch
else
Step 5: print"No extensions found"
[end od if]
Step 6: Stop
```

Source Code:

```
echo -n "Enter the filename : "
read f
for f in $*
do
        if [ -f $f ]
        then
                ch=$(echo $f | cut -d '.' -f2-)
                echo -n "$ch, "
        else
                echo -n  "No extension is found!!"
        fi
```

```
done
```

Output:

```
Enter the filename : a.txt
txt
```

Discussion: The main objective of the program is to show the usefulness of the cut command by which we have cut the delimeter and print the extension.

ASSIGNMENT NO. 12

PROGRAM STATEMENT:
Write a shell script to rename a group of files.

THEORY:
Unix normally stores file in different formats like *.sh,.txt,.cpp,*etc.We could simply use the "**cut**" command based on "." as the delimiter to convert various files to any other type.Mainly the extension is to be checked and changed.

ALGORITHM:
Steps:
Send the current extension and the target extension as command line arguments.
Find all the files(*)with the extension as sent as 1st argument.
Use "cut" command to cut the name of those specific files,concatenate them with the target or desired extension using simple string concatenation.
Move the original file to a new file with the newly created name.
End
Program Script(code):
```
Clear
if [ $# -eq 2 ]
then
f1="*."$1
f2=$2
for i in $f1
do
n=` echo $i|cut–d"."–f 1 `
n=$n.$f2
mv $i $n
done
else
echo –e "\nnumber of arguments wrong"
fi
```

INPUT:

```
$ls
```

OUTPUT:

```
dellinux.sh      ff        Music              series4747.sh      Videos
Desktop                 ff.txt    new                 Rimi           xxx.htm
division.sh    file.txt  Pictures              Templates      test.htm
Documents      Lnx.txt   positiveintrev.sh    type.sh
Downloads      marks     Public              userlogged
dualfilereading.sh  marks1    reverse.sh         userlogged.sh
```

INPUT:

```
$sh extenchang.sh htm html#(original script file)
```

OUTPUT:

```
dellinux.sh      ff        Music              series4747.sh      Videos
Desktop                 ff.txt    new                 Rimi   xxx.html
division.sh    file.txt  Pictures              Templates      test.html
Documents      Lnx.txt   positiveintrev.sh    type.sh
Downloads      marks     Public              userlogged
dualfilereading.sh  marks1    reverse.sh         userlogged.sh
```

Discussion:

In this way many file extensions can be changed.
The simplicity in this program is that Unix can easily change the extension thus allowing string concatenation.
Same file with different extensions may exists,so removing of one file is necessary.

ASSIGNMENT NO. 13

PROGRAM STATEMENT:
Write a shell script to find the number of word that containing vowel in a text file.

ALGORITHM:

Steps:

```
1. Ask user for an input file.
2. A) Check the file that is exist or not;then
        1) Check the file that there are how many vowels are exist.
        2) By using grep command we are checking the pattern
           [aeiouAEIOU] in that given input file.
   B) If the file does'nt exist then
        1) print "file doesn't exist"
3. End if.
```

PROGRAM SCRIPT(CODE):

```
echo -n "Enter the filename : "

read f

if [ -f $f ]

then

        echo -n "No. of words containing vowel : "

        grep  -c '[aeiouiAEIOU]' $f

else

        echo "File not found!!"

fi
```

INPUT:

```
cat > ab
mango
blackberry
dry
fly
mosquito
lunch

cat ab
mango
blackberry
dry
fly
mosquito
lunch
```

OUPUT:

```
bash p31.sh
Enter the filename : ab
No. of words containing vowel : 4
```

DISCUSSIONS:

1. First we must have to check the existancy of a given input file,without this we can’t conclude that whether the vowels are present In the content or not.
2. Here we use the ‘grep’ command to search for vowels in the contents of the file.

ASSIGNMENT NO. 14

PROGRAM STATEMENT:

Write a shell script to find a large name to its sorted representation.

ALGORITHM:

1) Ask user for the string.
2) Replace the space with newline using ‘tr’ command in the string to get the words.
3) Sort the word using ‘sort’ command.
4) Replace the space with newline using ‘tr’ command in the string to get the sorted representation.

Steps:

PROGRAM SCRIPT(CODE):

```
echo –n “Enter the string >> “
read str
echo –n “Name in sorted representation >> “
echo $str | tr  ‘ ‘ ‘ \n ‘ |tr ‘\n’ ‘ ‘
echo
```

INPUT:

```
$bash sort.sh
```

OUPUT:

```
Enter a string:Vinay Sen
Name in sorted representation:Sen Vinay
```

DISCUSSIONS:

The string is taken as input.The spaces of the file are replaced by newlines using ‘tr’ command’.The standard output of the command to the left of the pipe gets sent as standard input of the command to the right of the pipe.Sort command is then used to sort the words in the string.