Aim

To write simple shell scripts using shell programming fundamentals.

The activities of a shell are not restricted to command interception is alone. The shell also rudimentary programming features. when a group of commands has to be executed regularly they are stored in a file (.sh with extension). all such files are called shell scripts. or shell programs. shell programs can in interactive mode.

The original UNIX came with the Bourne shell and it is universal even today. Two of them c shell and korn shell has been well accepted. by unix fraternity. Linux offers Bash shell as a superior alternative to Bourne shell.

PRELIMINARIES

1. Comments in shell scripts start with '#'. It can be placed any where in the line the shell ignore contents to its right. comments are recommended but not mandatory.
2. Shell variables are loosely typed (ie) not declared Their type depends on value assigned Variable when used in an expression or output must be prefixed by $
3. The read statement is shell's internal tools for making script interactive.
4. output is displayed using echo statement. Any text should be within quotes escapes sequence. should be used in -e option.

5. Commands are always enclosed with ` ` (back quote)
6. Expressions are computed using the expr command. Arithmetic operators are +, -, *, /, %.
7. Multiple statements can be written in single line separated by ;
8. The shell scripts are executed using the sh command.

2.1A SWAPPING VALUES OF TWO VARIABLES.

algorithm.

Step1 : Start
Step2 : Read the value of a and b
Step3 : Interchange the values of a and b using another variable t as follows

```
t=a
a=b
b=t
```

Step4 : print a and b
Step5 : Stop.

Program

```
#swapping values.
echo -n "enter value for A:"
read a
echo -n "enter value for B:"
read b
t=$a
a=$b
b=$t
echo "values after swapping"
echo "A value is $a"
echo "B value is $b"
```

Algorithm.

Step1 : Start
Step2 : Read fahernheit Value
Step3 : convert fahernheit to centigrade using the formulae.
C fahernheit - 32) x 5/9
Step4 : point centigrade.
Step5 : Stop.

Program.

#degree conversion.

echo -n " enter fahernheit"
read f
c='expr \(($f - 32)\) * 5/9'
echo " centigrade is : $c"

2.1c AREA & CIRCUMFERENCE OF CIRCLE

Algorithm.

Step1 : Start
Step2 : define constant pi = 3.14
Step3 : Read the value for radius
Step4 : calculate area using formulae pi * radius^2.
Step5 : calculate circumference. using formulae 2 * pi * radius
Step6 : point area and circumference
Step7 : stop

Program.

# code metrices using read only variable
pi= expr "scale=2 ; 22/7" \bc
readonly pi # pi cannot be altered.
echo -n " Enter value for radius"
read radius
area = 'expr "scale=2; $pi * $radius * $radius" \
bc circum = 'expr "scale=2; 2* $pi * $radius" \
bc' echo - "Area : $area"
echo " circumference : $circum"

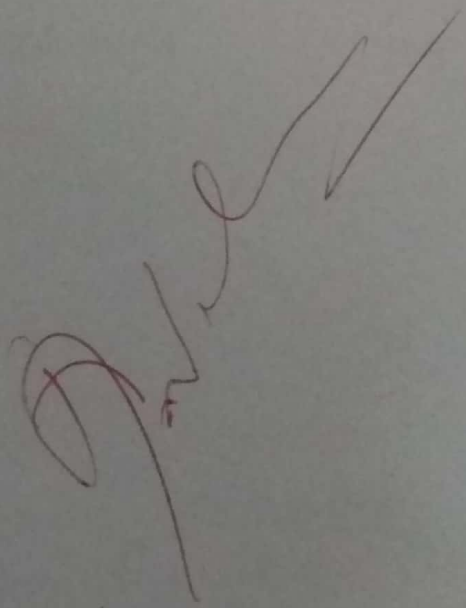
2.10 – SIMPLE INTEREST CALCULATION

Algorithm.

Step1: Start
Step2: Read the values principal and rate and years
Step3: compute simple interest using formulae: (P×rate×Y)/100
Step4: print simple interest
Step5: Stop

Program.

```
#interest computation using bc
echo -n "Enter principal amount"
read p
echo -n "Enter number of years:"
read n
echo -n "Enter rate of Interest:"
read r
si =`expr "scale=2; $p * $n * $r/100" |bc`
echo "Simple Interest : $si"
```



CONDITIONAL CONSTRUCTS

AIM

To write shell scripts using decision making constructs.

Shell supports decision-making using IF statement. The IF statement like its counterpart in programming language has following formats. The first construct executes the statements when the condition is true. The second construct adds an optional else to the first one that has different set of statements to be executed depending on whether condition is true or false. The last one is an elif ladder in which condition are tested in sequence, but only one set of statement is executed.

```
if [condition]
then
   statements
fi
```

```
if [condition]
then
   statements
else
   statements
fi
```

```
if [condition]
then
   statements
elif [condition]
then
   statements
...
else
   statements
fi
```

The set of relational and logical operators used in conditional expression is given below. The numeric operation in the shell is confined to integer value only.

operators	description
-eq	equal to
-ne	not equal to
-gt	greater than.
-ge	greaterthan or equal to
-lt	less than.
-le	less than or equal to
-a	logical AND
-o	logical OR
!	logical NOT

2-2A ODD OR EVEN

ALGORITHM

Step1: Start
Step2: Read number
Step3: If number divisible by 2 then
print "Number is Even"

Step3-1: else
print 'Number is odd"

Step4: stop.

Program.

#odd or even using if-else.
echo -n "Enter a non-zero number"
read num
rem = 'expr $num % 2'
if [$rem -eq 0] then.
echo "$num is Even" else.
echo "$num is odd"
fi.

2-2B BIGGEST OF 3 NUMBERS.

ALGORITHM

Step1: Start
Step2: Read value of a, b and c
Step3: If a>b and a>c then.
Print 'A is the biggest"
Step3-1: else if b>c then
Print "B is the biggest"
Step3-2: else
print "C is the biggest"
Step4: Stop.

Program (big3.sh)

```
# Biggest using logical expression.
echo -n "Give value for A, B and C:"
read a b c
if [ $a -gt $b -a $a -gt $c ] then
echo "A is the biggest number"
elif [ $b -gt $c ] then
echo "B is the biggest number" else
echo "C is the biggest number"
fi
```

2.2C - LEAP YEAR.

ALGORITHM

Step1: Start
Step2: Read the value of the year
Step3: If year divisible by 400 then.
Paint "Leap year"
Step3.1: else if year divisible by 4 and not divisible by 100 then.
Paint "leap year"
Step3.2: else
Step 4: stop

Program.

```
# leap year
echo "-n" Enter a year:"
read year
rem1 = 'expr $ year % 4'
rem2 = 'expr $ year % 100'
rem3 = 'expr $ year % 400'
if [ $rem3 -eq 0 ] then.
echo "$ year is a leap year"
elif [ $rem2 -ne 0 -a $ rem1 -eq 0 ]
then
echo "$ year is a leap year"
else.
echo "$ year is not a leap year"
fi.
```

P.3D - GRADE DETERMINATION

Algorithm.

Step1: Start

Step2: Read mark

Step3: If mark > 90 then
Print "S grade"

Step3-1: else if mark > 80 then
Print "A grade"

Step3-2: else if mark > 70 then
Print "B grade"

Step3-3: else if mark > 60 then
Print "C grade"

Step3-4: else if mark > 55 then
Print "D grade"

Step3-5: else if mark > 50 then
Print "E grade"

Step3-6: else
Print "U grade"

Step4: Stop.

Program.

```
echo -n "Enter the mark:"
read mark
if [$mark -gt 90] then
echo "S grade"
elif [$mark -gt 80] then
echo "A grade"
elif [$mark -gt 70] then
echo "B grade"
elif [$mark -gt 60] then
echo "C grade"
elif [$mark -gt 55] then
echo "D grade"
elif [$mark -gt 50] then
echo "E grade"
elif [$mark -gt
else
    echo "U grade"
fi.
```

2.2E - STRING COMPARISON

ALGORITHM

Step1 : Start
Step2 : Read strings str1 and str2
Step3 : If str1 = str2 then
        Print "strings are the same"
Step3.1 : else
        Print "string are distinct"
Step4 : Stop

PROGRAM

```
echo -n "Enter the first string:"
read s1
echo -n "Enter the second string:"
read s2
if [ $s1 == $s2 ] then
echo "strings are the same"
else
echo "strings are distinct"
fi.
```

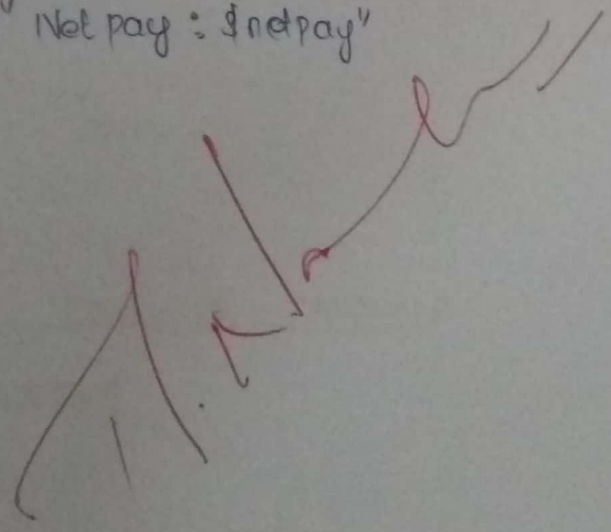
2.2F - Employee pay calculation.

ALGORITHM.

Step1 : Start
Step2 : Read basic
Step3 : If basic > 30000 then.
        hra is 5% of basic
        da is 5% of basic
        tax is 10% of basic.
Step3.1 : else if basic > 20000 then.
        hra is 4% of basic
        da is 3% of basic
        tax is 8% of basic.
Step3.2 : else
        hra is 3% of basic
        da is 2% of basic
        tax is 5% of basic.
Step4 : Stop.

Program:

```
echo -n " Enter employee basic pay :"
read basic
if [ $basic -gt 30000 ] then
    hra = `expr 5 \* $basic / 100`
    da = `expr 5 \* $basic /100`
    tax = `expr 10 \* $basic / 100`
elif [ $basic -gt 20000 ] then.
    hra = `expr 4 \* $basic /100`
    da = `expr 3 \* $basic /100`
    tax = `expr 8 \* $basic/100`
else
    hra = `expr 3 \* basic /100`
    da = `expr 2 \* basic / 100`
    tax = `expr 5 \* basic / 100`
fi
gross = `expr $basic + $da + $hra`
netpay = `expr $gross - $tax`
echo "gross pay : $gross"
echo " Net pay : $netpay"
```



RESULT :

MULTIWAY BRANCHING

AIM:

To write shell scripts using case construct to match patterns.

The case statements is used to compare a variable value against a set of constants. If it matches a constant then the set of statements is executed till a ;; is encountered. The optional default block is indicated by *. Multiple constants can be specified in a single pattern separated by |.

```
case Variable
  in constant(1)
     statements
  ;; constant2)
     statements ;;
  ...
  (constantN)
   statements ;; *)
   statements
  esac.
```

2.3A - VOWEL OR CONSONANT

ALGORITHM

Step1: Start
Step2: read char
Step3: If char is either 'a', 'e', 'i', 'o' or 'u' then.
        Print "It's a vocal"
Step3.1: else
        Print "It's a Consonant"
Step4: stop.

Program.

```
#read with multiple values in a pattern
echo -n "key in a lower case character :"
read choice
case $choice in.
    a|e|i|o|u)
echo "It's a vowel" ;; *)
echo "Its a consonant"
esac.
```

2.3.B SIMPLE CALCULATOR.

ALGORITHM

Step1: Start
Step2: read operands a and b
step3: display operation menu
Step4: Read option.
Step5: If option = 1 then.
calculate c = a+b
Step5.1: else if option=2 then
calculate c=a-b
Step5.2: else if option=3 then
calculate c = a*b
Step5.3: else if option=4 then
calculate c= a/b
Step5.4: else if option=5 then
calculate c = a%b
Step5.5: else
step6: print c.
step7: stop.

Program.

```
# Arithmetic operations - multiple statements in a block echo
echo -n "Enter the two numbers:"
read a b
echo "1. Addition"
echo "2. Substraction"
echo "3. Multiplication"
echo "4. division"
echo "5. modulo division"
echo -n "Enter the option:"
read option.
case $option in
1) c='expr $a + $b'
   echo "$a + $b = $c" ;;
2) c='expr $a - $b'
   echo "$a - $b = $c" ;;
3) c='expr $a \* $b'
   echo "$a \* $b = $c" ;;
4) c='expr $a / $b'
   echo "$a \* $b = $c ;;
5) c='expr $a % $b' echo
   echo "$a % $b = $c" ;;
*) echo "Invalid Option"
esac.
```

Looping.

AIM.

To write shell scripts using loop statements.

shell supports a set of loops such as for, a while and until to execute a set of statements repeatedly. The body of the loop is contained between do and done statement.

The for loop doesn't test a condition, but uses a list instead.

```
for Variable in
list do
    statements
done.
```

The while loop executes the statements as long as the condition remains true.

```
while [condition] do
statements
done.
```

The until loop complements the while construct in the sense that the statements are executed as long as condition remains false.

```
until [condetion]
do
statements
done.
```

2.4A MULTIPLICATION TABLE.

ALGORITHM.

Step1: Start

Step2: read the value of n.

Step3: Initialize 1 to i

Step4: Print n, i, n x i

Step5: increment i by 1

Step6: repeat step 4 and 5 until i 10
Step7: stop.

Program.

```
#multiplication table using for loop class.
echo -n " which multiplication table? "
read n.
for x in 1 2 3 4 5 6 7 8 9 10
do
 p=`expr $x \* $n`
echo -n "$n x $x = $p"
Sleep 1
done.
```

2.4.B AMSTRONG NUMBER.

ALGORITHM.

Step1: start
Step2: read number
Step3: initialize 0 to sum and number to num.
Step4: extract last digit by computing number modulo 10
Step5: cube the last digit and add it to sum.
Step6: divide number by 10
Step7: Repeat step 4-6 until number >0
Step8: If sum = number then
        print "Amstrong number"
Step9: else
        print "not a amstrong"
Step10: stop.

Program.

```
loop  echo -n "Enter a number:"
read  n
a=$n
s=0
while [$n -gt 0]
do
  r=`expr $n %10`
  s=`expr $s + \($r \* $r \* r \)
  n=`expr $n/10`
```

```
done
if [$a -eq $s]
then
  echo "Amstrong number"
else
  echo -n " not an amstrong number"
fi.
```

2.4.c number reverse.

Algorithm.

Step1: Start
Step2: read number
Step3: Initialize 0 to reverse
Step4: Compute rev = rev*10 + last digit.
Step5: print reverse.
Step6: stop.

Program.

```
loop echo -n "enter a number"
read n
rd=0
while [$n -gt 0] do
rem='expr $n%10'
rd='expr $rd \*10 + $rem'
n='expr $n/10'
done
echo "reversed number is $rd".
```

2.4.D FIBONACCI SERIES

ALGORITHM

Step1: Start
Step2: read number of terms as n.
Step3: Initialize 0 to f1, 1 to f2 and 2 to i
Step4: print initial fibonacci terms f1, f2.
Step5: generate next term using the formula f1+f2.
Step6: printf3
Step7: increment i by 1.

Step8: Assign f2 to f1
Step9: assign f3 to f2
Step10: Repeat steps 5-9 until i n
Step11: Stop.

Program.

```
loop echo -n "enter number of terms"
read n.
echo "fibonacci series" f1=0 f2=1
echo -n "$f1"
echo -n "$f2" i=2
while [$i -lt $n]
do
  f3 = 'expr $f1 + $f2
  'echo -n "$f3" f1 = $f2
   f2 = $f3
   i= expr $i+1'
   done
```

PRIME NUMBER

ALGORITHM.

Step1: Start
Step2: Read the Value n
Step3: Initialize i to 2
Step4: If n is divisible by i then print "not prime" and stop
Step5: Increment i by 1
Step6: Repeat step 4 and steps until i n/2
Step7: print "prime"
Step8: Stop.

Program.

```
echo -n "enter the number:"
read n.
i=2
m= 'expr $n/2'
until [$i -gt $m]
do
 q= 'expr $n % $i'
```

```
if [ $q -eq 0 ] then.
echo " not a prime number " exit
fi
i=`expr $i+1` done
echo " prime number "
```

2.4 FACTORIAL VALUE.

ALGORITHM.

step1: start
step2: read number
step3: initialize 1 to fact and number to i
step4: fact = fact * i
step5: decrement i by 1
step6: repeat step 4 - 6 until i>0
step7: print fact
step8: stop.

PROGRAM.

```
echo -n "enter a positive number"
read n
f=1
until [ $n -lt 1 ] do
 f= `expr $f \* $n`
 n= `expr $n - 1`
done
echo " factorial value : $f"
```

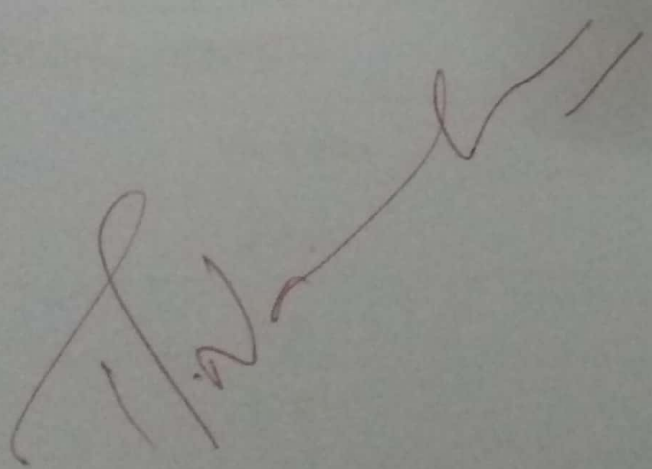
2.4.9 SUM OF 1...N NATURAL NUMBER.

ALGORITHM.

step1: start
step2: read n.
step3: initial 0 to sum & to i
step4: add i to sum
step5: increment by i
step6: print sum
step7: stop.

Program:

```
echo -n "enter N value"
read n
sum=0
i=1
until [$i -gt $n] do
sum='expr $sum + $i'
i='expr $i + 1'
done
echo "The sum of n number is $sum"
```



COMMAND BASED SCRIPTS.

AIM.

To write shell scripts using shell commands.

FILE TEST OPERATION.

condition	returnValue.
-e filename	True if file exist
-f filename	true if file exist and is ordinary.
-d filename	true if file exist and is directory.
-r filename	true if file exist and is readable
-w filename	true if file exist and is writable.
-x filename	true if file exist and is executable
-s filename.	true if file exist and is non-empty
file1 -nt file2.	true if file exist and is file2.

SPECIAL VARIABLES.

Variables	description.
$0	name of script
$n	command line arguments
$#	number of arguments
$?	exit status.

2.5.A TIME BASED GREETING.

Program (
#!/bin/bash.
x='date +%H'
mrd='date +%p' if
[$mrd == 'am']
then
if [$x -le 11]
then
echo "good morning"
fi
ebe
if[$x -le 2] then.
echo "good afternoon"
elif [$x -le 6]
then
echo "good evening"
ebe
echo "good night"
fi
fi

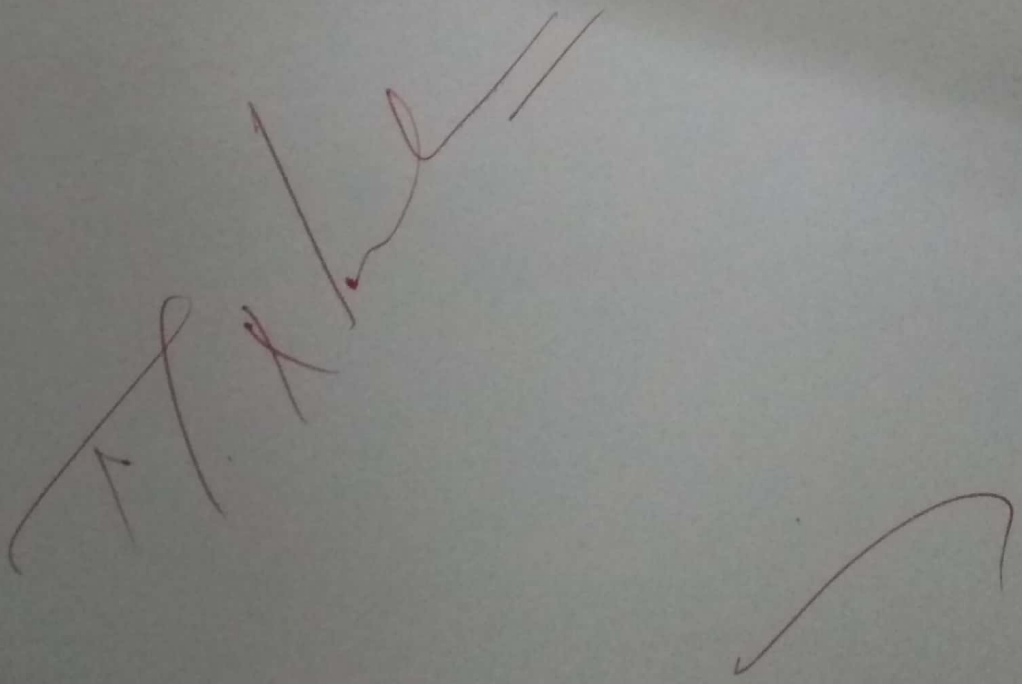
2.5B number of days in month.

month mth='date +%om'
mn='date +%B'
Case $mth in.
02) echo "febraary usually has 28 days"
echo "If leap year then if has 29 days" ;;
04|06|09|11)
echo "The cussent month $mn has 30 days" ;; *).
echo "The cussent month $mn has 31 days"
esac.

2.5C commands menu.

ch='y'
while [$ch == 'y']do.
echo -e "\t menu.
1. list of files
2. working directorly.

3. date and time
4. user of the system.
5. calender
enter the option:
"c"
read choice.
case "$choice" in.
1) ls -l ;;
2) pwd ;;
3) date ;;
4) who ;;
5) cal ;;
esac.
echo -n " do you wish to continue (Y/N)";
read ch
done



Result

Thus programming constructs using shell command