NAME: **Mahesh Vala**
SUBJECT: **UNIX and Shell Programming**
Roll No: **B-52**

Shell Scripts

1) Write a shell script to scans the name of the command and executes it. Script:

```
echo "Enter command name"
read cmd
$cmd
```

O/P:-

```
Enter command name
cal
February 2016
Su Mo Tu We Th Fr Sa
      1 2 3 4 5 6
 7 8 9 10 11 12 13
 14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29
```

2) Write a shell script Which works like calculator and performs below operations Addition , Subtract ,Division ,Multiplication.

Script :

```
i="y"
while [ $i = "y" ]
          do
        echo " Enter one no."
        read n1
        echo "Enter second no."
        read n2
        echo "1.Addition"
        echo "2.Subtraction"
        echo "3.Multiplication"
        echo "4.Division"
        echo "Enter your choice"
        read ch
        case $ch in
                1) sum=`expr $n1 + $n2`
                echo "Sum ="$sum;;
                2)sum=`expr $n1 - $n2`
                echo "Sub = "$sum;;
                3)sum=`expr $n1 \* $n2`
                echo "Mul = "$sum;;
                4)sum=`expr $n1 / $n2`
                echo "Div = "$sum;;
                *)echo "Invalid choice";;
        esac
        echo "Do u want to continue ? y/n"
        read i
        if [ $i != "y" ]
        then
                exit
        fi
done
```

O/P :

```
Enter one no.
32
Enter second no.
22
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
2
Sub = 10
Do u want to continue ? y/n
N
```

3) Write a shell script to find the largest among the 3 given numbers. Script :

```
clear
echo "Enter first number: "
read a
echo "Enter second number: "
read b
echo "Enter third number: "
read c
if [ $a -ge $b -a $a -ge $c ]
then
        echo "$a is largest integer"
elif [ $b -ge $a -a $b -ge $c ]
then
        echo "$b is largest integer"
elif [ $c -ge $a -a $c -ge $b ]
then
        echo "$c is largest integer"
fi
```

O/P:

```
Enter first number:
22
Enter second number:
33
Enter third number:
42
44 is largest integer
```

4) Write a shell script to reverse a number supplied by a user. Script:

```
if [ $# -eq 1 ]
then
```

```
        if [ $1 -gt 0 ]
        then
                num=$1
                sumi=0
                while [ $num -ne 0 ]
                do
                        lnum=`expr $num % 10`
                        sumi=`expr $sumi \* 10 + $lnum`
                        num=`expr $num / 10`
                done
                echo "Reverse of digits is $sumi of $1"
        else
                echo " Number is less than 0"
        fi
else
        echo "Insert only one parameter "
fi
```

<u>O/P:</u>

```
bash pr81.sh 123
Reverse of digits is 321 of 123
```

5) Write a shell script to count number of digits, vowels and cosonants. <u>Script:</u>

```
echo -n "Enter a line of text: "
read string

numCount=$(echo $string | grep -o "[0-9]" | wc --lines)
vowCount=$(echo $string | grep -o -i "[aeiou]" | wc --lines)
consCount=$(echo $string | grep -o -i "[bcdfghjklmnpqrstvwxyz]" | wc --lines)

echo "The given string has $vowCount vowels, $consCount consonants and $numCount
numbers in it."
```

<u>O/P:</u>

```
Enter a line of text: eeva sh1
The given string has 3 vowels, 3 consonants and 1 numbers in it.
```

6) Write a shell script to check whether the number is palindrome or not. <u>Script:</u>

```
echo -n "Enter a number: "
read num

# store the original number
original_num=$num

# reverse the number
rev=0
while [ $num -gt 0 ]
do
```

```
    # get the remainder of the number
    remainder=$(($num % 10))

    # multiply reverse by 10 then add the remainder
    rev=$((($rev * 10) + $remainder))

    # divide the number by 10
    num=$(($num / 10))
done

# check if the number is a palindrome
if [ $original_num -eq $rev ];
then
    echo "$original_num is a palindrome number."
else
    echo "$original_num is not a palindrome number."
Fi
```

O/P:

```
Enter a number: 121
 121 is a palindrome number.
```

7) Write a shell script to reverse a string.

Script:

```
echo "Enter a string : "
read s
strlen=${#s}
for (( i=$strlen-1; i>=0; i-- ));
do
    revstr=$revstr${s:$i:1}
done
echo "Original String : $s"
echo "Reversed String : $revstr"
```

O/P:

```
Enter a string :
reverse string
Original String : reverse string
Reversed String : gnirts esrever
```

8) Write a shell script to display name and size of the files on the given path. **Script:**

```
echo "Enter the full path to the file :"
read file
filesize=$(ls -lh $file | awk '{print $5 " " $9}')
echo "$file has a size of $filesize"
```

O/P:

```
Enter the full path to the file :
/home/hp/
```

```
/home/hp/ has a size of
70 cmb_file.txt
 1.7K cmb_file1.txt
210 cmb_file2.txt
39 cmp1.txt
```

9) Write a menu driven shell script to create and delete a file which will accept two command line arguments (file name and create / delete option).

<u>Script:</u>

```
case $1 in
        "--create")
                echo "Creating new file $2"
                #echo
                touch $2
                ;;
        "--delete")
                echo "Deleting file $2"
                echo
                rm $2
                ;;
        *)
                echo "Not a valid argument"
                echo
                ;;
esac
```

<u>O/P:</u>

```
$ bash egcase.sh --create f1.txt
Creating new file f1.txt
```

10) Write a shell script to count number of lines words and characters of a string and of a file.

<u>Script:</u>

```
echo -n "Enter a String : "
# Taking input from user
read text

# Counting words

word=$(echo -n "$text" | wc -w)
echo "No of Word :"$word
# Counting characters
char=$(echo -n "$text" | wc -c)

echo "no of char :"$char
# path to the file
file_path="/home/hp/demo.txt"
```

```
# using wc command to count number of lines
number_of_lines=`wc --lines < $file_path`

# using wc command to count number of words
number_of_words=`wc --word < $file_path`

# Displaying number of lines and number of words
echo “File name : $file_path”
echo "Number of lines: $number_of_lines"
echo "Number of words: $number_of_words"
```

O/P:

```
Enter a String : count characters
No of Word :2
no of char :16
File name : /home/hp/demo.txt
Number of lines: 17
Number of words: 16
```

10) Write a shell script to which represents the ways to declare and access array.

Script:

```
# To declare static Array
arr=(prachi poonam 1 richa ronak roocha)

# To print all elements of array
echo ${arr[@]}
echo ${arr[*]}
echo ${arr[@]:0}
echo ${arr[*]:0}

# To print first element
echo ${arr[0]}
echo ${arr}

# To print particular element
echo ${arr[3]}
echo ${arr[1]}

# To print elements from a particular index
echo ${arr[@]:0}
echo ${arr[@]:1}
echo ${arr[@]:2}
echo ${arr[0]:1}
```

```
# To print elements in range
echo ${arr[@]:1:4}
echo ${arr[@]:2:3}
echo ${arr[5]:1:3}

# Length of Particular element
echo ${#arr[3]}
echo ${#arr}

# Size of an Array
echo ${#arr[@]}
echo ${#arr[*]}

# Search in Array
echo ${arr[@]/*[aA]*/}

# Replacing Substring Temporary
echo ${arr[@]//a/A}
echo ${arr[@]}
echo ${arr[0]//r/R}
```

O/P:

```
prachi  poonam  1  richa  ronak  roocha
prachi  poonam  1  richa  ronak  roocha
prachi  poonam  1  richa  ronak  roocha
prachi  poonam  1  richa  ronak  roocha
prachi
prachi
richa
poonam
prachi poonam 1 richa ronak roocha
poonam 1 richa ronak roocha 1 richa
ronak roocha
rachi
poonam 1 richa ronak
 1 richa ronak
ooc
5
6
6
6
1
prAchi poonAm 1 richA ronAk roochA
prachi poonam 1 richa ronak roocha
pRachi
```

11) Write a shell script to convert a binary number to decimal number. Script:

```
# Take input as binary number
echo "Enter Binary Number -"
read n

# function to convert binary to decimal number
function binaryCon(){

    local i=0
    local num=0

    # while loop
    while [ $n != 0 ]
    do
    digit=`expr $n % 10`
    num=$(( num + digit * 2**i ))
    n=`expr $n / 10`
    (( ++i ))
    done

    # print the resultant decimal number
    echo "Resultant Decimal Number"
    echo "$num"
}

# Function Call
binaryCon
```

O/P:

```
Enter Binary Number
101
Resultant Decimal Number
5
```

12) Execute commands for below listed tasks.
Create a file named eg_grep.sh. Write the content related to UNIX in the same and use that file to perform following command.

- **a) Display list of all the files which have word “UNIX” in it.** `$grep -l "UNIX" *`
- **b) Search for the patter “UNIX” in a file and display the lines which does not have the given pattern.**
  `$grep -v "UNIX" eg_grep.txt`
- **c) Display the lines of a file which ends with “labs.”**
  `$grep "labs.$" eg_grep.txt`
- **d) Parenthesize first letter of such words which have first capital letter in that word.**
  `$sed 's/\(\b[A-Z]\)/\(\1\)/g' eg_grep.txt`

e) Duplicate the line in which string/word is replaced.

```
$sed 's/is/IS/p' eg_grep.txt
```

f) Delete 2 to 4 line of the given file.

```
$sed '2,4d' eg_grep.txt
```

13)Create a file named emplyee.txt. Add employee details(employee name, designation, department and salary) in that file. Perform below given tasks on that file.

a) Display line number in front of each line.

```
$awk '{print NR,$0}' employee.txt
```

b) Display row number and name separated by '-'.

```
$awk '{print NR "- " $1 }' employee.txt
```

c) Display the length of the longest line.

```
$awk '{ if (length($0) > max) max = length($0) } END { print max }' employee.txt
```

d) Display record of the employees whose designation is "clerk". $awk '{
if($2 == "clerk") print $0;}' employee.txt

GREP COMMAND

1) **-c Displaying the count of number of matches**: prints only a count of the lines that match a pattern
grep -c "unix" eg_grep.txt

2) **-i Case insensitive search:** Ignores, case for matching grep -i "UNix" eg_grep.txt

3) **-l Display the file names that matches the pattern:** Displays list of a filenames only.
grep -l "is" *

or

$grep -l "is" index.txt op.txt sort1.txt eg_revstr.sh

4) **-n Show line number while displaying the output using grep:** Display the matched lines and their line numbers. grep -n "unix" eg_grep.txt

5) **-v Inverting the pattern match:** This prints out all the lines that do not matches the pattern
grep -v "unix" eg_grep.txt

6) **-f file :** Takes patterns from file, one per line.
grep -f pattern.txt eg_grep.txt

7) **-w Checking for the whole words in a file:** Match whole word grep -w "unix" eg_grep.txt

8) **-o Displaying only the matched pattern:** Print only the matched parts of a matching line, with each such part on a separate output line.
grep -o "unix" eg_grep.txt

9) **Matching the lines that start with a string:**
grep "^unix" eg_grep.txt
10) **Matching the lines that end with a string : $** grep "labs.$" eg_grep.txt

-A n : Prints searched line and nlines after the result. grep -A1 learn eg_grep.txt

-B n : Prints searched line and n line before the result. grep -B1 learn eg_grep.txt

-C n : Prints searched line and n lines after before the result. grep -C1 learn
eg_grep.txt

SED COMMAND

Replacing or substituting string :

sed 's/operating system/os/' eg_grep.txt

Replacing the nth occurrence of a pattern in a line : sed 's/is/IS/2'

eg_grep.txt

Replacing all the occurrence of the pattern in a line : flag /g (global replacement)

sed 's/is/IS/g' eg_grep.txt

Replacing from nth occurrence to all occurrences in a line : sed 's/is/IS/2g'

eg_grep.txt

Parenthesize first character of each word :

sed 's/\(\b[A-Z]\)/\(\1\)/g' eg_grep.txt

Replacing string on a specific line number :
sed '3 s/one/ONE/' eg_grep.txt

Duplicating the replaced line with /p flag :
sed 's/is/IS/p' eg_grep.txt

Printing only the replaced lines :
sed -n 's/is/IS/p' eg_grep.txt

Replacing string on a range of lines :

replaces the lines with range from 1 to 3

sed '1,3 s/is/IS/p' eg_grep.txt

replaces the text from second line to last line

sed '3,$ s/is/IS/p' eg_grep.txt

Delete a particular line say n in this example

sed '2d' eg_grep.txt

To Delete a last line
sed '$d' eg_grep.txt

To Delete line from range x to y sed '2,4d'
eg_grep.txt To Delete from nth to last line sed
'3,$d' eg_grep.txt To Delete pattern matching
line sed '/uNix/d' eg_grep.txt

AWK COMMAND

1. Default behavior of Awk: By default Awk prints every line of data from the specified file.

```
awk '{print}' employee.txt
```

2. Print the lines which match the given pattern.

```
awk '/manager/ {print}' employee.txt
```

3. Splitting a Line Into Fields :

```
awk '{print $1,$4}' employee.txt
```

Built-In Variables In Awk

NR built-in variables (Display Line Number)

```
awk '{print NR,$0}' employee.txt
```

NR built-in variables (Display Line From 3 to 6)

```
awk 'NR==3, NR==6 {print NR,$0}' employee.txt
```

NF built-in variables (Display Last Field)

```
awk '{print $1,$NF}' employee.txt
```

To print the first item along with the row number(NR) separated with ” – “

```
awk '{print NR "- " $1 }' employee.txt
```

To print any non empty line if present

```
awk 'NF < 0' employee.txt
```

To find the length of the longest line present in the file:

```
awk '{ if (length($0) > max) max = length($0) } END { print max }'  employee.txt
```

To count the lines in a file:

```
awk 'END { print NR }' employee.txt
```

Printing lines with more than 10 characters:

```
awk 'length($0) > 29' employee.txt
```

To find/check for any string in any specific column:

awk '{ if($2 == "clerk") print $0;}' employee.txt **To print the squares of first numbers from 1 to n say 6:** awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i*i; }'