Assignment for O.S. Lab (Sem 5)

A1. Welcome to Bash learning and **********on separate line. (Video 478)

Welcome to Bash learning

and **********

on seperate line

#!/bin/bash

My first Bash script

echo "Hello World"

echo $SHELL

echo `date`

http://spoken-tutorial.org; http://sakshat.ac.in.

A2. Write a simple Bash program to get the following system variables pwd logname. (Video 479)

#!/bin/bash

username=sunita

echo "outside function: $username"

func()

{

```
echo "inside function: $username"

}

func
```


A3. Write a simple Bash program To ask username from user

To exit the program, if user does not enter anything within 10 seconds Hint: read -t 10 –p. (Video 479)

```
#!/bin/bash

username=sunita

echo "outside function: $username"

func()

{

local username=jack

echo "inside function: $username"

}

func
```

A4. Using wildard , redirect all the files starting from letter 'c' to another file name' Myfiles'. (Video 480)

#!/bin/bash

echo "zeroth arg: $0"

==========

Type:

echo "first arg: $1"

echo "second arg: $2"

echo "third arg: $3"

==========

Type:

echo "twelveth arg: ${12}"

==========

Type:

```
echo "total args: $#"
==========
Type:

echo "1st $* "
echo "Args(dollar *): $*"
echo "2nd  $*"
for arg in "$*"
do
 echo "$arg"
done
==========
Type:

echo "1st $@  "
echo "Args(dollar @): $@"
for arg in "$@"
```

do

echo "$arg"

done

A5. Write a bash program for addition using command line arguments. (Video 480)

A6. Write a Bash script to do all operations discussed under Globbing. (Video 481)

A7. Declare Array names of length 7 and find (Video 482)

a) The total number of elements.

b) Print all the elements.

c) Print the 5th element.

#!/bin/bash

declare -a Linux=('Debian' 'Redhat' 'Ubuntu' 'Fedora')

echo -e "Total number elements in array Linux: ${#Linux[@]} \n"

echo -e "The elements of array Linux are: ${Linux[@]} \n"

```
echo -e "Third element in array Linux is: ${Linux[2]} \n"

echo -e "Length of third element is: ${#Linux[2]} \n"
```

A8. Declare an Array names 2 of length 7 and perform following operations. (Video 483)

```
#!/bin/bash


declare -a Linux=('Debian' 'Redhat' 'Ubuntu' 'Fedora')


echo -e "Original elements in an array Linux: ${Linux[@]} \n"

echo -e "The two elements starting from index one(Redhat): ${Linux[@]:1:2}\n"


Linux[2]='Mandriva'

echo -e "All elements after replacement: ${Linux[@]} \n"
```

```
Linux=("${Linux[@]}" "Suse")

echo -e "All elements After appending Suse: ${Linux[@]} \n"


unset Linux[2]

echo -e "All elements after removal of third element: ${Linux[@]} \n"
```

A9. Extract three elements starting from index two. Replace third element with 'XXX' and display. (Video 483)

A10. Append any new name at the end of Array. (Video 483)

A11. Write a script: (Video 484)

Take your name as an input

It should check this name with your system's username

If the username match, it should greet you by displaying Hello

Else, it should display Try again

Hint: Your system's username is stored in a variable $USER.

```
#!/bin/bash

PASS="abc123"
read -s -p "Enter password: " mypassword
if [ "$mypassword" == "$PASS" ];
then
    echo -e "\nPassword accepted"
else
    echo -e "\nAccess denied"
fi
#!/bin/bash

count=100

if [ $count -eq 100 ]; then
    echo "Count is 100"
fi
```

A12. Write a program to output different messages: (Video 485)

1) When number is greater than 3.

2) Lesser than 3.

3) Equal to 3 or when user input is empty.

```
#!/bin/bash

read -p "Enter a word : " string

if [ -z "$string" ]; then
    echo "Nothing was entered "

elif [[ "$string" == *"raj"* ]]; then
    echo "\"$string\" contains word 'raj'"

elif [[ $string = *"jit"* ]]; then
    echo "\"$string\" contains word 'jit'"

else
       echo "Sorry! entered word does not contain either 'raj' or 'jit'"
```

```
fi
#!/bin/bash

NAME="anusha"
PASSWORD="abc123"

read -p "Enter name: " myname
if [ "$myname" = "$NAME" ]; then
    read -s -p "Password: " mypassword
    if [ "$mypassword" = "$PASSWORD" ]; then
      echo -e "\nWelcome"
    else
      echo -e "\nWrong password"
    fi
else
    echo "Wrong name"
fi
```

A13. Check whether the file exists and is executable using logical operators. (Video 502) [Hint: man test]

```
#!/bin/bash
read -p "Enter a Word : " string
if [ -z "$string" ]; then
    echo "Nothing was entered "

elif [[ "$string" == *"raj"* ]] && [[ "$string" == *"jit"* ]]; then
     echo "$string contains both the words 'raj' and 'jit'"

elif [[ "$string" == *"raj"* ]] || [[ $string = *"jit"* ]]; then
    echo "$string contains the word 'raj' or 'jit'"
else
    echo "Sorry! The entered word '$string' does not contain either 'raj' or 'jit'"
fi
#!/bin/bash

if [ ! -f "$1" ]; then
```

```
    echo "File '$1' does not exist"
else
    echo "File '$1' exists"
fi
```

A14. 1) Write a program to demonstrate the use of not equal to operator. (Video 503)Hint: -ne.

```
#!/bin/bash
echo "Enter filename: "
read y
x=`cat $y | wc -w`
if [ $x -eq 0 ]; then
echo "$y has zero words"
fi
if [ $x -ne 0 ]; then
echo "$y has $x words"
fi
#!/bin/bash
# checks for number of characters in a file
```

```
echo "Enter the filename: "
read y
x=`cat $y | wc -c`

if [ $x -lt 1 ] ; then
echo "No characters present in $y"
fi



 if [ $x -gt 1 ] ; then
echo "$y has more than one character"

        if [ $x -ge 1 ] && [ $x -le 100 ]; then
          echo "Number of characters ranges between
1  and 100"
        fi

        if [ $x -gt 100 ] ; then
```

```
        echo "Number of characters is above 100"
       fi
fi
```

A15. Explore some more attributes: (Video 504)

```
-r
#!/bin/bash

file1=/home/ttt/fileattrib.sh
# file1="/etc/mysql/debian.cnf"

if [ -f $file1 ];
then
    echo "File exists and is a normal file"
else
    echo "File does not exist"
fi



if [ -s $file1 ];
```

```
then
    echo "File exists and is not empty"
else
    echo "File is empty"
fi

if [ -w $file1 ];
then
    echo "User has write permission to this file"
else
    echo "User doesn't have write permission to this file"
fi

-x
#!/bin/bash

file1="/home/ttt/empty1.sh"
file2="/home/ttt/empty2.sh"
```

```
if [ $file1 -nt $file2 ];
then
    echo "file1 is newer than file2"
else
    echo "file2 is newer than file1"
fi

if [ $file1 -ot $file2 ];
then
    echo "file1 is older than file2"
else
    echo "file2 is older than file1"
fi
-o
#!/bin/bash

if [ "$(whoami)" != 'root' ];
then
```

```
    echo "You have no permission to run $0 as non-root user."
else
    echo "Welcome root!. "
    exit 0
fi
```

A16. Find the sum of first n prime numbers. (Video 505).

```
#!/bin/bash

read -p "Enter a number: " y
sum=0

for ((i=1;i<=y;i++))
do
    echo $((sum=sum+i))
done
echo "Sum of first n numbers is $sum"
```

```
#!/bin/bash

for file in $(ls -1);
do
      echo $file;
done
#!/bin/bash

read -p "Enter a number:" number
i=0
sum=0

    while [ $i -le $num ]
    do
      echo $((sum=sum+i))
      i=$[$i+2]
    done
      echo "Sum of even numbers within the given range is $sum"
```

A17. Retype nested-for.sh bash script using nested while loop

Save our program with the name: nested-while. Sh. (Video 528)

```
#!/usr/bin/env bash


for dir in test*; do
    echo "Files in $dir directory:"
    echo ""
    for file in $(ls -1 $dir); do
      echo $file
    done
    echo "-----------------------"
done
```

A18. Write a menu driven program for mathematical calculation. (Video 529)

It should take user inputs a and b.

It should ask for mathematical operator (+, -, / and *).

Do the calculation and print the output.

```
#!/usr/bin/env bash


space=`df -h | sort -rk5 | awk 'FNR == 2 {print $5}' | cut -d "%" -f1`


case $space in

   [0-6][0-9]) echo "Everything is OK"

     ;;

    [7-8][0-9] | 9[1-8]) echo "Clean out. There's a partition that is $space % full."

     ;;

    99) echo " Hurry. There's a partition at $space %!"

     ;;

     *) echo "This is nonexistent amount of disk space..."

     ;;

esac
```

A19. Write a program with two functions. (Video 542)

a) The first function should display disk space usage in human readable form.

(Hint: df -h).

b) The second function should display file system usage in human readable form.

(Hint: du -h).

```
#!/usr/bin/env bash


function machine #Function declaration
{
   # Beginning of Functin definition
   echo -e "\nMachine information:" ; uname -a
   echo -e "\nUsers logged on:" ; w -h
   echo -e "\nMachine status :" ; uptime
   echo -e "\nMemory status :" ; free
   echo -e "\nFilesystem status :"; df -h
   # End of Function definition
}
```

```
echo “Beginning of program”
machine      #Its a function call
echo “End of program”
```

A20. Write a program, where the function accepts two arguments. (Video 546)

```
#!/usr/bin/env bash

say_hello () {

   local first_name=$1    # Local variable
   last_name=$2           # Global variable

          echo "Hello $first_name $middle_name $last_name"
}

middle_name="K"          #global variable
```

```
say_hello "Pratik" "Patil"


echo "My first name is: $first_name"

echo "My middle name is: $middle_name"

echo "My last name is: $last_name"
```

A21. The function should multiply the two arguments. (Video 546)

```
#!/usr/bin/env bash


say_welcome () {


    echo "Welcome to $1 $2"

}


say_welcome "Bash" "learning"

say_welcome "functions in" "Bash"
```

A22. Make 3 function calls with arguments - (1, 2), (2, 3) and (3, 4). (Video 546)

A23. Write a program (Video 559)

a) Where a function adds all the elements in an array.
b) #!/usr/bin/env bash
c)
d) array_display() {
e)
f) array=($@) # Array name as variable
g) echo "Array elements are: ${array[@]}" # Displays all element of an array
h) echo "Second element is: ${array[1]}" # Displays the 2nd element
i) }
j)
k) operating_systems=(Ubuntu Fedora Redhat Suse)
l) array_display ${operating_systems[@]}
m)
n) colors=(White green red blue)
o) array_display ${colors[@]}

b) The function should display the sum of elements.

#!/usr/bin/env bash

```
function return_function () {
    if [ $1 == $2 ]; then
      echo "This is return function"
      return 0 # Returns to main program
      echo "This will not appear"
    fi
}



function exit_function () {
    if [ $1 == $2 ]; then
      echo "This is exit function"
      exit 0
    fi
}

return_function 3 3
echo "We are in main program"
```

```
exit_function 3 3

echo "This line is not displayed"
```

c) Make 2 function calls with array elements- (1, 2, 3) and (4, 5, 6).

A24. Write a function add to add two numbers and call the function in another file. (Video 567)

```
#!/usr/bin/env bash


bg_function() {
    echo -e "Inside bg_function\n"
    find . -iname "*.mp3" > myplaylist.txt
}
```

```
#!/usr/bin/env bash
```

```
function machine
{
echo "function machine is called in function.sh file"
}
```

A25. Write a program where the recursive function calculates the sum of N numbers. (Video 568)

```
#!/usr/bin/env bash

#Recursive factorial function
factorial() {
    echo "We are inside a factorial function"
}

# Main script
read -p "Enter the number: " n

if [ $n -eq 0 ];
```

```
then
    echo "factorial value of $n is 1"
else
    # Calling factorial function
    factorial $n
fi
#!/usr/bin/env bash
#Recursive factorial function

factorial() {
    echo "We are inside a factorial function"
}

# Main script
read -p "Enter the number:" n
if [ $n -eq 1 ];
then
    echo "factorial value of 0 is 1"
else
```

```
    # Calling factorial function
    factorial $n
fi
```

Type below code snippet inside a 'factorial' function replacing the

echo statement.

Replace 'echo "We are inside a factorial function"' statement with the

following code.

```
    temp=$1
    if [ $temp -eq 1 ]; then
      echo "1"
    else
      f=$((temp-1))
```

```
    #Recursive call
    f=$(factorial $f)
    f=$((f*temp))
    echo $f
  fi
```

A26. Write a program in any language like C, C++, Java.

And redirect the output or error to a new file. (Video 570)

#!/usr/bin/env bash

OR

Create a text file with some content like your name, address.

Redirect the content to a new file.

rm: cannot remove `/tmp/4815.txt': No such file or directory.

9

8

7

0

3

2

A27. Create X file.txt files with some content. (Video 582)

ls: cannot access /user: No such file or directory

/usr:

bin

games

include

lib

local

sbin

share

src

Thu Jul 31 11:18:32 IST 2014

A28. Redirect the content of both out file.txt and X file.txt to a new file. (Video 582)

#!/usr/bin/env bash

 # Redirection of standard output and error to file "out_file.txt"

ls /usr /user &> out_file.txt

A29. Convert a string to uppercase using: (Video 583)

a) Here document

```
#!/usr/bin/env bash
wc -w << HERE
   Hello
   and
   Welcome
   to
   Bash
   learning
HERE

wc -w <<< 'Welcome to Bash learning'
```

b) Here string
c)#!/usr/bin/env bash
d)
e) cat << this
f) 0'th argument is: $0
g) 1st argument is: $1
h) 2nd argument is: $2
i) this

(Hint: tr a-z A-Z).

A30. Try to append few lines to a file test.txt using file descriptor. Display the content of the file using file descriptor. (Video 603)

```
#!/usr/bin/env bash

exec 3> output.txt

echo "Welcome to BASH learning" >&3

date >&3

echo "Hi" >&3

exec 3<&-
```

```
#!/bin/bash
exec 3< output.txt
cat <&3
# Close fd # 3
exec 3<&-
```

Compiled & Prepared by

Navin Kumar