# Script for Week 8 (a)

a) Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else

```
echo "Enter file/directory"
read str
if test -f $str
        then echo "file exists and it is ordinary file"
        elif test -d $str
        then echo "It is Directory"
        else
                echo "Else something"
fi
```

# Script for Week 8 (b)

b) Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory.

```
 # Check if at least one argument is provided
if [ $# -eq 0 ]; then
      echo "Usage: $0 filename1 filename2 ... is required "
      exit 1
fi
# Loop through all the arguments
for file in "$@"; do
# Check if the file exists in the current directory
      if [ -f "$file" ]; then
# Convert the file content to uppercase and overwrite the file
            tr '[:lower:]' '[:upper:]' < "$file" > tmpfile && mv tmpfile "$file"
            echo "Converted $file to uppercase."
      else echo "File $file does not exist in the current directory."
      fi
done
```

# Script for week 8 (c)

c) Write a shell script that determines the period for which a specified user is working on the System.

```
username=$1
# Check if the user is currently logged in
current_login=$(who | grep "^$username\s")
if [ -z "$current_login" ]; then
    echo "User $username is not currently logged in."
    else
    echo "User $username is currently logged in."
    echo "$current_login"
fi
# Check the user's last login/logout sessions
echo "Last login sessions for user $username:"
last $username | head -n 10
```

# Script for week 9 (a)

Program to implement

a) Write a shell script to perform the following string operations:

i) To extract a sub-string from a given string.

ii) To find the length of a given string.

```
echo "Enter any string"
read str
n=${#str}
echo $n
echo "Enter the start position of substring"
read s1
echo "Enter the end position of substring"
read f1
echo $str | cut -c $s1-$f1
```

9 b) Write a shell script that accepts a file name starting and ending line numbers as arguments and displays all the lines between the given line numbers.

```
# Check if the correct number of arguments are provided
if [ $# -ne 3 ]; then
  echo "Usage: $0 Enter filename start_line end_line"
  exit 1
fi
# Assign arguments to variables
file=$1
snum=$2
enum=$3
# Display the lines between the given line numbers
sed -n "$snum','$enum' 'p" $file
```

10 a) Write a shell script that computes the gross salary of a employee according to the following rules:

i) If basic salary is < 1500 then HRA =10% of the basic and DA =90% of the basic.

ii) If basic salary is >=1500 then HRA =Rs500 and DA=98% of the basic

The basic salary is entered interactively through the key board.

```
echo "enter the basic salary:"
read bsal
if [ $bsal -lt 1500 ]
then
        gsal=$((bsal+((bsal/100)*10)+(bsal/100)*90))
        echo "The gross salary : $gsal"
fi
if [ $bsal -ge 1500 ]
then
        gsal=$(((bsal+500)+(bsal/100)*98))
        echo "the gross salary : $gsal"
fi
```

10 b) b) Write a shell script that accepts two integers as its arguments and compute the value of first number raised to the power of the second number.

```
echo "Enter the integer value :"
read int1
echo "Enter the power of that integer:"
read int2
power=$int1
i=1
while [ $i -lt $int2 ]
do
power=`expr $power \* $int1`
i=`expr $i + 1`
done
echo "The value of first number=$int1 to the power of the second
number=$int2 is $power "
```

11) Write an interactive file-handling shell program. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, then program ask the user for the necessary information, such as the file name, new file name

```
echo "Welcome to the File Handling Program"
echo "Please choose an option:"
echo "1. Copy a file"
echo "2. Remove a file"
echo "3. Rename a file"
echo "4. Create a symbolic link to a file"
echo "5. Exit"
read -p "Enter your choice [1-5]: " choice
case $choice in
1) read -p "Enter the source file name: " source
     read -p "Enter the destination file name: " destination
if cp "$source" "$destination"; then
echo "File copied successfully."
else
echo "Error: Failed to copy the file." fi
..
```

11) Write an interactive file-handling shell program. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, then program ask the user for the necessary information, such as the file name, new file name

```
2) read -p "Enter the file name to remove: " file
   if rm "$file"; then
      echo "File removed successfully."
      else
      echo "Error: Failed to remove the file."
   fi
   ;;
3) read -p "Enter the current file name: " old_name
   read -p "Enter the new file name: " new_name
   if mv "$old_name" "$new_name"; then
   echo "File renamed successfully."
   else echo "Error: Failed to rename the file."
   fi
   ;;
```

11) Write an interactive file-handling shell program. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, then program ask the user for the necessary information, such as the file name, new file name

```
4) read -p "Enter the target file name: " target
   read -p "Enter the symbolic link name: " link_name
   if ln -s "$target" "$link_name"; then
   echo "Symbolic link created successfully."
   else
   echo "Error: Failed to create symbolic link."
   fi
   ;;
5) echo "Exiting the program."
exit 0
;;
*) echo "Invalid choice. Please run the program again."
exit 1
;;
esac
```

12) a) Write shell script that takes a login name as command – line argument and reports when that person logs in

```
# Check if the login name is provided as an
argument
if [ "$#" -ne 1 ]; then
echo "Usage: $0 <login_name>"
exit 1
else
login_name=$1
last $1 && echo"Details of user $1"
fi
```

12 b) Write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

```
echo "enter the first file name"
   read file1
echo "enter the second file name"
  read file2
cmp $file1 $file2 && rm $file2
if [-e $file1 ] then
       if [ !-e $file2 ] then
         echo " the two files contents are
same. so $file2 is deleted"
            else
        echo " the two file contents are not
same and $file2 not deleted"
      fi
else
     echo "$file1 is not existed"
fi
```

12 b) Write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

```
if [ "$#" -ne 2 ]; then
echo "Enter: $0 file1_name file2_name"
exit 1
fi
file1=$1
file2=$2
# Check if both files exist

if [ ! -f "$file1" ] || [ ! -f "$file2" ]; then
echo "Both files must exist."
exit 1
fi

# Compare the two files
if cmp -s "$file1" "$file2"; then
echo "The files are same, So second file $file2 is deleted"
rm "$file2"
else
echo "The files are different."
fi
```

13 a) Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions.

```
echo "The list of file is"
for file in *
do
        if [  -f $file ]
        then
                if [ -r $file -a -w $file -a -x $file ]
                then
                        ls -l $file
                fi
        fi
done
```

13 b) Develop an interactive script that asks for a word and a file name and then tells how many times that word occurred in the file.

```
echo "Enter the word to be searched"
read word
echo "Enter file name"
read file
echo "the number of time the word "$word" is occurred in file
$file is "
grep -o $word $file | wc -l
```