OPERATING SYSTEM ASSIGNMENT 1

Question 1: Write a shell script that has 2 user created variables, uv1 and uv2. Ask for the values of the variables from the user and take in any values (real/integer/character) for the 2 variables. Test the program for different types of uv1 and uv2. (a) Print them as: (i) value of uv1 followed by value of uv2 separated by a comma and (ii) value of uv2 followed by value of uv1 separated by the word "and". (b) Print the variables in reverse order [If uv1 is 1234, then output should be 4321].

APPROACH: The approach is quite simple. I will take inputs of uv1 and uv2 using read command. Then, I will print them according to the instructions given in the question using the echo command.

For reversal of each variable, firstly, length of each input will be calculated, then they are printed character-by-character from last to first.

Code Snippet:

```
#! /bin/bash

echo "Enter uv1 and uv2 values: "
read uv1 uv2

echo $uv1 , $uv2
echo $uv2 and $uv1

# FINDING LENGTH OF VARS
len1=${#uv1}
len2=${#uv2}

echo -n "uv1 in reverse is: "

for (( i=$len1-1; i>=0; i-- )) # PRINTING IN REVERSE
do
    echo -n ${uv1:i:1}
done

echo

echo -n "uv2 in reverse is: "

for (( i=$len2-1; i>=0; i-- )) # PRINTING IN REVERSE
do
    echo -n ${uv2:i:1}
done
```

Question 2: Write a shell script to count the number of lines in a file. Test if the file is present. If not, create and write.

APPROACH: Firstly, for the checking if file is present or not, -e is used. Again, if the file is not present, a new file is created using touch command and anything the user enters is written within the file using echo.

For the count of lines in the file, the command wc -l is used.

Code Snippet:

```
#! /bin/bash

read -p "Enter filename: " filename

if [ -e $filename ] # CHECKING IF FILE EXISTS OR NOT
then
    echo "File exists..."
else
    echo "File doesn't exist.. creating..."
    touch $filename
    echo "Enter something to write to the file"
    read data
    echo $data > $filename
fi

# no of lines of file
echo -n "Number of lines in the file: "
wc -l < $filename
echo
```

Question 3: Write a shell script that counts the number of ordinary files (not directories) in the current working directory and its sub-directories. Repeat the count of files including the sub-directories that the current working directory has.

APPROACH: For this question a function is created that checks whether a passed filename is an ordinary file or directory. If it is a directory, then recursive call is made again for that subdirectory, otherwise the ordinary filename is printed on the screen. For listing of filenames within a directory for in loop is used.

Code Snippet:

```
#! /bin/bash

# RECURSIVE FUNCTION TO LIST ALL THE FILES
# IN CURRENT DIRECTORY AND SUB-DIRECTORIES
listFiles(){
    for file in *
    do
        if [ -d $file ] # IF IT IS A DIRECTORY THEN CALL RECURSIVELY
        then
            cd $file
            listFiles $1/$file
            cd ..
        else
            echo $1/$file
        fi
    done
}

# PASSING THE CURRENT DIRECTORY AS ARGUMENT
listFiles "."
```

Question 4: Write a shell program to duplicate the UNIX rm command with the following features: a. Instead of deleting the files, it will move them to a my-deleted-files directory. If the file already exists in the my-deleted-files directory, then the existing file (in the my-deleted-files) will have the version number zero (0) appended to it and the newly deleted file will have version number one (1) appended to it. Go on incrementing the version nos., if required. b. The command will have a switch -c that will clear the entire my-deleted-files directory after asking for confirmation.

APPROACH: For solving this question, we must follow some steps. Firstly, similar to rm I created a function remove. This function first checks whether -c switch passed or not, if yes, then the recycle bin is completely deleted, then remove is called again for the argument file without -c. In normal case, without the -c switch, if recycle bin is not present then created using mkdir. Then a function is there that keeps on counting files with same name present in recycle bin (ex- test.txt, test(1).txt). After the completion of counting, the parameter file is moved to recycle bin with necessary sequence number (ex – test(2).txt).

Code Snippet:

```bash
#! /bin/bash
recycle_bin="my-deleted-files"

countIt(){ #counts number of versions present already in recycle bin
            # and return the final to be inserted version
    vari=$1
    flnm=""
    extn=""
    strlen=${#vari}

    for (( c=0; c<$strlen; c++ )); do
    if [ ${vari:c:1} = "." ]; then
        extn+=${vari:c:($strlen-$c+1)}
        break
    fi
    flnm+=${vari:c:1}
    done

    if [ ! -e $flnm$extn ]; then
        echo $vari
        return
    fi
    cnt=1
    while true
    do
        if [ ! -e "$flnm($cnt)$extn" ]; then
            echo "$flnm($cnt)$extn"
            return
        fi
        let "cnt++"
    done

}
remove(){
    if [ $1 = "-c" ]; then #recycle bin will be cleared
        rm -r $recycle_bin
        remove $2
        return
    fi

    if [ -e $1 ]; then
        if [ ! -e $recycle_bin ] # no recycle bin directory
        then
            mkdir $recycle_bin
            echo "Recycle bin created"
        fi
        cd $recycle_bin
        loc=$(countIt $1)
        cd ..
        mv "./$1" "./$recycle_bin/$loc"
        echo "Moved to recycle bin"
    else
        echo "File not exists.."
    fi
}
read -p "Enter a filename: " file
remove $file
```

Question 5: **Write a script called birthday_match.sh that takes two birthdays of the form DD/MM/YYYY (e.g., 15/05/2000) and returns whether there is a match if the two people were born on the same day of the week (e.g., Friday). And then find out the age/s in years/months/days.**

APPROACH: There is date command, which I will use to solve this question. Firstly, two birthdays are taken as input from the user in the mentioned format. Then each birthday is converted to an UNIX format timestamp. Later, the day of week is extracted as string from that UNIX date using +%A. Whether two dates are of same day of week can be easily checked now.

Code Snippet:

```
#! /bin/bash

takeDate(){
    user_date=$1

    # EXTRACTING DAY MONTH AND YEAR
    dd=${user_date:0:2}
    mm=${user_date:3:2}
    yy=${user_date:6:4}

    # ECHOING IN UNIX FORMAT
    date -d "$yy-$mm-$dd" +%A
}

# TAKING THE TWO BITHDAYS
read -p "Enter first bithday: " date1
read -p "Enter second bithday: " date2

# TAKING THE DAY OF THE WEEK FROM THE FUNCTION
day1=$(takeDate $date1)
day2=$(takeDate $date2)

# CHECKING WHETHER BOTH DAYS ARE SAME OR NOT
if [ $day1 = $day2 ]; then
    echo "wow . same day - $day1"
else
    echo "Different day - $day1 and $day2"
fi
```

Question 6: **Write a shell script that accepts a file name as an input and performs the following activities on the given file. The program asks for a string of characters (that is, any word) to be provided by the user. The file will be searched to find whether it contains the given word. If the file contains the given word, the program will display (a) the number of occurrences of the word. The program is also**

required to display (b) the line number in which the word has occurred and no. of times the word has occurred in that line (Note: the word may occur more than once in a given line). If the file does not contain the word, an appropriate error message will be displayed.

APPROACH: Using grep and wc command we can find out the frequency of string within a file. If frequency is zero then an error message is displayed, otherwise total frequency is displayed and, using a loop frequency per line is also displayed.

Code Snippet:

```
#! /bin/bash

read -p "Enter Filename: " filename
read -p "Enter Word to Search in file: " string

echo
freq=$(grep -o $string $filename | wc -l)
if [ $freq == 0 ]
then
    echo "No \"$string\" found in file \"$filename\""
    exit 1
fi

echo "Frequency of occurence of word '$string' in file '$filename' is
 : $freq"

freqarr=(`grep -o -n $string $filename | cut -d : -f 1 | uniq -c`)

echo
echo "Table showing the line wise frequencies of '$string'"
echo "-----------------------------------------------------"
echo -e "\tLine number \tFrequency"
echo "-----------------------------------------------------"

for(( i=0; i<${#freqarr[@]}; i+=2 )); do
    echo -e "\t${freqarr[$i+1]} \t\t${freqarr[$i]}"
done
echo "-----------------------------------------------------"
echo
```

Question 7: Extend the shell script written in (6) to perform the following task: User is asked to enter two different patterns or words. The first pattern will have to be matched with the contents of the file and replaced by the second pattern if a match occurs. If the first pattern does not occur in the file, an appropriate error message will be displayed

APPROACH: If the word to be replaced count is greater than 0 then, using sed command we will replace all the instances of that word with the new word given by the user.

Code Snippet:

```bash
#!/bin/bash

read -p "Enter the file path :- " file_path
read -p "Enter the search string :- " search
read -p "Enter the replace string :- " replace

word_count=$(grep -o -n $search $file_path | wc -l)

if [ "$word_count" -eq 0 ]; then
    echo "Word Not Found !"
    exit
fi

if [[ $search != "" && $replace != "" ]]; then
    sed -i "s/$search/$replace/g" $file_path
fi
```