pattern_1.sh

```
<<DOC
Name: Babu Malagaveli
Date: 26.04.2023
Description: A1 - Read 'n' and generate a pattern given below( number increments from left to right)Virtual programming lab
Sample Input: 4
Sample Output:
1
1 2
1 2 3
1 2 3 4
DOC
read -p "Enter the limit: " limit
if [ $limit -gt 0 ]
then
for row in `seq $limit`
do
for col in `seq $row`
do
echo -n "$col "
done
echo
done
else
echo "Error: Invalid Input"
fi
```

pattern_2.sh

```
#!/bin/bash
<<DOC
Name: Babu Malagaveli
date: 26.04.2023
description: Read 'n' and generate a pattern given below(number increasing from Top to bottom)
sample input: 4
sample output:
1
2 3
4 5 6
7 8 9 10
DOC
#!/bin/bash
number=1                        #declaring the initial number as 1
read n                          #reading the input from the user
for i in `seq $n`               #looping along the rows till n
do
    for j in `seq $i`           #looping across columns after each row
     do
        echo -n "$number "       #printing the number
        number=$((number+1))     #incrementing the each number by 1
    done
    echo
done

```

arithmatic_calc.sh

```
Name: Babu Malagaveli
Date: 26.04.2023
Description: Write a script for arithmetic calculator using command line arguments
Sample Input: ./arithmatic_calc.sh 25 + 41
Sample Output: 25 + 41 = 66 and other operations
DOC
#!/bin/bash
# Switch Case to perform
# calulator operations
if [ $# = 0 ]                                                   #if the number of CLI is 0 then throw an error
then
    echo "Error : Please pass the arguments through command line."
    echo "Usage:./arithmatic_calc.sh 2.3 + 6.7"
elif [ $# = 3 ]                                                #if the number of CLI is 3 then proceed for case statement
then
case $2 in
   +)res=`echo $1 + $3 | bc`;;
   -)res=`echo $1 - $3 | bc`;;
   x)res=`echo $1 \* $3 | bc`;;
   /)res=`echo "scale=2; $1 / $3" | bc`;;
esac
echo "$1 $2 $3 = $res"
else                                                            #if the no if CLI is less/more than 3 , throw an error
    echo "Error:Please pass 3 arguments."
    echo "Usage:./arithmatic_calc.sh 2.3 + 6.7"
fi

```

operator_dependent.sh

```
<<DOC
Name: Babu Malagaveli
Date: 29.04.2023
Description: script to perform arithmetic operation on digits of a given numberVirtual programming lab
Sample Input: 1236+
Sample Output: 12
DOC
#!/bin/bash
var=$1                                                          #initializing the cla to var including the opr
res=${var:0:1}                                                  #initializing the first digit of the number to res
opr=${var: -1}                                                  #getting the last letter of the number
length=${#var}                                                  #finding the length of the digit given  by the user
#echo $length
if [ $# -gt 0 ]                                                 #if anything given in the cla then proceed
then
for((i=1;i<$length-1;i++))                                      #iterating through the each digit of the number using loop
do
#echo ${var:i:1}
case $opr in
+)res=$((res + ${var:i:1}));;                                   #using the loop of the ith iteration , performing the operation with case statement
-)res=$((res - ${var:i:1}));;
x)res=$((res * ${var:i:1}));;
/)res=$(echo "scale=2;$res / ${var:i:1}" | bc);;
*)echo "Error: Operator missing or invalid operator, please pass operator as last digit (+,-,x,/)";;
esac
done

case $opr in                                                    #then finally after the loop ends, we're printing the appropriate case
+)echo "Sum of digits = $res";;
-)echo "Subtraction of digits = $res";;
x)echo "Multiplication of digits = $res";;
/)echo "Division of digits = $res";;
*);;
esac
else
echo "Error : Please pass the arguments through CL."                #if the user didn't provide any argument through command line the print error
echo "Usage : ./operator_dependent.sh 2345+"
fi
```

string_length.sh

```
<<DOC
Name: Babu Malagaveli
Date: 27.04.2023
Description: A6 - Write a script to print the length of each and every string using arrays
Sample Input: ./string_length.sh hello hai how are you?
Sample Output:
Length of string (hello) - 5
Length of string (hai) - 3
Length of string (how) - 3
Length of string (are) - 3
Length of string (you?) - 4
DOC
#!/bin/bash
array=($@)
len=${#array[@]}
args=${array[@]:0:$len}

if [ $len -eq 0 ]
then
    echo "Error : Please pass the arguments through command-line."
else
for((i=o;i<$len;i++))
do
    echo "Length of string (${array[@]:i:1}) - ${#array[i]}"
done
fi

```

```
<<DOC
Name: Babu Malagaveli
Date: 29.04.2023
Description: A8 - Write a script to sort a given number in ascending or descending order
Sample Input:./sorting.sh -a 5 4 6 2 3 8 9 7 1
Sample Output:
Ascending order of array is 1 2 3 4 5 6 7 8 9
#Using Bubble Sort technique tp solve this problem
DOC
#!/bin/bash
array=($@)
arr=(${array[@]:1})                                     #Initially initializing the CLI with the first element
len=${#arr[@]}                                          #finding the length of the array
if [ $# -gt 1 ]
then
    case $1 in
        -a)
            for((i=0;i<=len-1;i++))                     #looping through all the digits based on the len of the array
            do
                for((j=0;j<=len-2;j++))                     #looping through all the digits based on the len of the array less 1
                do
                    if [ ${arr[j]} -gt ${arr[j+1]} ]        #if the previous element is gt the next element
                    then
                        temp=${arr[j]}
                        arr[j]=${arr[j+1]}                  #swapping the elements
                        arr[j+1]=$temp
                    fi
                done
            done
            echo "Ascending order of array is ${arr[@]}";;          #printing the sorted array
        -d)
            for((i=0;i<=len-1;i++))                         #looping through all the digits based on the len of the array
            do
                for((j=0;j<=len-2;j++))                     #looping through all the digits based on the len of the array less 1
                do
                    if [ ${arr[j]} -lt ${arr[j+1]} ]        #if the previous element is gt the next element
                    then
                        temp=${arr[j]}
                        arr[j]=${arr[j+1]}                  #swapping the elements
                        arr[j+1]=$temp
                    fi
                done
            done
            echo "Descending order of array is ${arr[@]}";;
            *)
            echo "Error : Please pass the choice."          #if the user gives neither -a or -d then printing error
            echo "Usage : ./sorting -a/-d 4 23 5 6 3"

        esac

    else
            echo "Error : Please pass the argument through command line."
            echo "Usage : ./sorting -a/-d 4 23 5 6 3"
fi

```

system_info.sh

```
<<DOC
Name: Babu Malagaveli
Date: 03.05.2023
Description: Write a script to print system information using commands
Sample Input: y and choice 2
Sample Output: Your shell directory is /bin/bash
DOC



#!/bin/bash
choice=y                                                #initilay storing the user choice as yes so that it will enter the loop
while [ $choice == y ]                                  #if the user choice is 'y', then stay inside the loop
do
    echo "1. Currently logged users
2. Your shell directory
3. Home directory
4. OS name & version
5. Current working directory
6. Number of users logged in
7. Show all available shells in your system
8. Hard disk information
9. CPU information.
10. Memory Informations
11. File system information.
12. Currently running process."

read -p "Enter the choice :" ch                         #reading the user choice
case $ch in
    1)whoami ;;                                         #Currently logged users.
    2)echo $SHELL ;;                                    #Your shell directory.
    3)echo cd ~ ;;                                      #Home directory.
    4)uname -sr ;;                                      #OS name & version.
    5)pwd ;;                                            #Current working directory.
    6)who -q ;;                                         #Number of users logged in.
    7)cat /etc/shells ;;                                #Show all available shells in your system.
    8)hwinfo ;;                                         #Hard disk information.
    9)cat /proc/cpuinfo ;;                              #CPU information.
    10)cat /proc/menuinfo ;;                            #Memory information.
    11)df/du ;;                                         #File system information.
    12)ps ;;                                            #Currently running process.
    *) echo "Error : Invalid option, please enter valid option"
esac
read -p "do you want to continue y/n : " choice`        #termination condition if the user choose 'n'.
done


```

Submitted on Friday, 28 April 2023, 3:40 PM (Download)

file_upper_lower.sh

```
<<DOC
Name: Babu Malagaveli
Date:28.04.2023
Description: A10 - Write a script to rename a file/directory replaced by lower/upper case letters
Sample Input:$ ls
File.txt MyScript.SH MyFile007.txt dir/ Assign1/ newfolder/
Sample Output: $ ls
file.txt myfile007.txt myscript.sh DIR/ ASSIGN1/ NEWFOLDER/
DOC
#!/bin/bash
for i in `ls`                                           #looping through ls(contains all dirs and files)
do
    if [ -f $i ]                                        #checking if it is a file
    then
        fvar=`echo $i | tr '[:upper:]' '[:lower:]'`     #if yes then translating all files to lower case
        if [ $i != $fvar ]                              #since we're inside a loop making sure the operated file is not changed again
        then
            mv $i $fvar                                 #and moving it to the fvar at runtime
        fi
    elif [ -d $i ]                                      #and the same process applies to dirs as well
    then
        dvar=`echo $i | tr '[:lower:]' '[:upper:]'`
        if [ $i != $dvar ]
        then
            mv $i $dvar
        fi
    fi
done
ls                                                     #at the end listing all contents inside a directory


```

Submitted on Friday, 28 April 2023, 4:13 PM (Download)

rename_album.sh

```
<<DOC
Name: Babu Malagaveli
Date:28.04.2023
Description:A11 - Given album name and corresponding directory, this scripts renames the jpg files with new name passed through command line

Sample Input:
DSN001.jpg DSN002.jpg DSN003.jpg DSN004.jpg DSN005.jpg

Sample Output:
 All .jpg files in current directory is renamed as
day_out001.jpg day_out002.jpg day_out003.jpg day_out005.jpg day_out004.jpg
DOC
#!/bin/bash
if [ $# -gt 0 ]
then
    echo `ls *.jpg`
for i in `ls *.jpg`
do
    var=`echo $i | tr -cd [:digit:]`
        mv $i $1$var.jpg
    done
    echo "All .jpg files in current directory is renamed as"
    echo `ls *.jpg`
 else
        echo "Error : Please pass the prefix name through command line."
       fi
~
```

Submitted on Friday, 28 April 2023, 10:34 PM (Download)

print_lines.sh

```
<<DOC
Name: Babu Malagaveli
Date: 28.04.2023
Description: Write script to print contents of file from given line number to next given number of lines.
Sample Input:
./print_lines.sh 5 3 myfile.txt
Sample Output:
line number 5
line number 6
line number 7
DOC
#!/bin/bash
var=`cat $3 | wc -l`                                        #storing count of the number of lines in the file given by the user
var1=$(($1+$2-1))                                           #storing count of the number from where to print and till where - 1
#echo $var1
if [ $# -eq 3 ]                                            #if the user provides the 3 arguments only
then
if [ $var -gt $var1 ]                                       #if the number of lines in the file provided is -gt the no to be printed
then
     head -$var1 $3 | tail -$2                              #from the file starting from $2 to $var1 we are printing the lines
else
    echo "Error: data.txt is having only $2 lines. file should have atleast $var1 lines."
fi
else
    echo "Error: arguments missing!"                          #if the user didnt provide the 3 arguments in the CLI , print error
    echo "Usage: ./file_filter.sh start_line upto_line filename"
    echo "For eg. ./file_filter.sh 5 5 <file>"
fi


```

Submitted on Friday, 5 May 2023, 12:25 PM (Download)

largest_username.sh

```
<<DOC
Name: Babu Malagaveli
Date: 05.05.2023
Description: Display the longest and shortest user-names on the system
Sample Input: ./largest_username.sh
Sample Output:
The Longest Name is: speech-dispatcher
The Shortest Name is:lp

DOC
#!/bin/bash

username=(`cat /etc/passwd | cut -d ":" -f1`)                   #using the filter commands cut command and piping storing all the usernames in a variable
short_username=${username[0]}                                   #initializing the first username from the list of usernames to a variable to compare to compare for short one
long_username=${username[0]}                                    #initializing the first username from the list of usernames to a variable to compare for long one
for i in ${username[@]}                                         #now looping therough the list if usernames one by one
do
    if [ ${#i} -gt ${#long_username} ]                          #checking if each iterations username is longest or not as compared to to prevoius one and storing in long_username

    then
        long_username=$i
    fi
    if [ ${#i} -lt ${#short_username} ]                         #checking if each iterations username is shortest or not as compared to to prevoius one and storing in short_username
    then
        short_username=$i
    fi
done
echo "The Longest Name is: $short_username"                     #finally print the long and short usernames from the directory /etc/passwd
echo "The Shortest Name is: $Long_username"

```

say_hello.sh

```
<<DOC
Name: Babu Malagaveli
Date: 29.03.2023
Description:Write script called say_hello, which will print greetings based on time
Sample Input: When we start shell (whenever you opening new tab or terminal)
Sample Output:
Good Morning user, Have nice day!
This is Thursday 08 in June of 2017 (10:44:10 AM)
DOC
#!/bin/bash
time=(`date | cut -d " " -f4`)                                  #fetching the time from the date command (HH:MM:SS)
day=(`date | cut -d " " -f1`)                                   #fetching the day from the date command
month=(`date | cut -d " " -f2`)                                 #fetching the month alone
year=(`date | cut -d " " -f6`)                                  #getting the year alone
hour=(`date | cut -d " " -f4 | cut -d ":" -f1`)                 #fetching the hour alone (HH)

if [ $hour -ge 5 -a $hour -lt 12 ]                              #as per the time wrote the conditional statements
    then                                                        #but here i used the 24 hr format
        echo "Good Morning `whoami`, Have a nice day!"
        echo "This is $day$date in $month of $year ($time AM)"
elif [ $hour -ge 12 -a $hour -lt 13 ]
    then
        echo "Good Noon `whoami`, Have a nice day!"
        echo "This is $day$date in $month of $year ($time PM)"
elif [ $hour -ge 13 -a $hour -lt 17 ]
    then
        echo "Good afternoon `whoami`, Have a nice day!"
        echo "This is $day$date in $month of $year ($time PM)"
elif [ $hour -ge 17 -a $hour -lt 21 ]
    then
        echo "Good evening `whoami`, Have a nice day!"
        echo "This is $day$date in $month of $year ($time PM)"

else
        echo "Good night `whoami`, Have a nice day!"
        echo "This is $day$date in $month of $year ($time)"
fi


```

upper_lower.sh

```
<<DOC
Name: Babu Malagaveli
Date:27.04.2023
Description:To convert string lower to upper and upper to lower
Input :
1. ./upper_lower.sh file.txt
1 – Lower to upper
2 – Upper to lower
Please select option : 1
Output:
WHAT ARE THE DIFFERENT OS?
WHEN IS OS USED?
WHAT IS PARTITION AND ITS USE?
HOW MANY PARTITIONS CAN BE DONE?
DOC

#!/bin/bash
#check for argument count
if [ $# -eq 1 ]                                         #checking for command line aeguments
    then
    if [ -f $1 ]                                        #checking whether it is a file or not
    then
    if [ -s $i ]                                        #checking for the file contents
    then
    #echo "1-LOWER TO UPPER"
    #echo "2-UPPER TO LOWER"
    read -p "Please select option :" option             #reading the choice from the user and storing in option

case $option in
    1)
       cat $1 | tr [:lower:] [:upper:]                  #if user says 1 then converting to upper case
       ;;
    2)
       cat $1 | tr [:upper:] [:lower:]                  #if user says 2 then converting to lower case
       ;;
    *)
       echo "Error: No contents inside the file."       #even after the input from user , if no content in file printing error
       ;;
esac
fi
fi
 else
 echo "Error : Please pass the file name through command line."   #and the final case , if the command line is empty , passing error
fi

```

recursion.sh

```
<<DOC
Name: Babu Malagaveli
Date: 03.05.2023
Description: Use a recursive function to print each argument passed to the function
Sample Input:
./recursion.sh How are you? I am fine

Sample Output:
How
are
you?
I
am
fine
DOC
#!/bin/bash
if [ $# -gt 0 ]                                    #if the user provides input via command line then proceed
then
function display()                                  #defining the function
{
arr=($@)                                            #initially storing the command line arguments into variable arr
echo $1                                             #printing the first element from the display $@ command since echo inside the function takes it
arr=(${arr[@]:1})                                   #eliminating the first element every time it is printed using the offset method
if [ ${#arr[@]} -gt 0 ]                             #checking if atleast one element is there in the arr
then
display ${arr[@]}                                   #using recursive func call within the function
fi
}
display $@                                          #calling the function
else
echo "Error : Pass the arguments through command line."       #if the user doesn't provide the command line arguments then throw an error
fi
```

mounted_fs.sh

```
<<DOC
Name: Babu Malagaveli
Date: 04.05.2023
Description:  Write a script to determine whether a given file system or mount point is mounted
Sample Input: ./mounted_fs.sh /dev/sda2
Sample Output:  File-system /dev/sda2 is mounted on / and it is having 98%
used space with 3298220 KB free.
DOC
#!/bin/bash
mounted=(`df | tr -s " " | cut -d " " -f6`)                                 #storing all the mounted on paths in a variable
file_name=(`df | tr -s " " | cut -d " " -f1`)                               #storing all the filenames in a variable array using tr and cut
used_space=(`df | tr -s " " | cut -d " " -f5`)                              #similarly storing all the used spaces in a variable
available_space=(`df | tr -s " " | cut -d " " -f4`)                         #and also stroing the available space in a variable
length=${#file_name[@]}                                                     #Calculating the length based on how many filenames are mounted in the system
flag=0
if [ $# -eq 0 ]                                                            #if the user doesn't provide the file system in the CLA , throw an error
then
    echo "Error : Please pass the name of the file-system through command line."
elif [ $# -eq 1 ]                                                           #if the user gives the file system in thr CLI , then proceed for the loop
then
    #for((i=1;i<=$length-1;i++))
    for i in `seq 1 $((length - 1))`                                        #loop with i as the iterative variable upto length-1
    do
        if [ "${file_name[$i]}" == $1 ]                                     #suppose if the filename given by the user is availabe in the array if file systems we stored then proceed
        then
        #printing the required details for the success case
            echo -e "file system $1 is mounted on ${mounted[$i]} and it is having ${used_space[$i]}\nused space with ${available_space[$i]} KB free"
            flag=1                                                          #turning the flag on for the success case
        fi
    done
    if [ $flag -eq 0 ]                                                      #printing the not mounted statement for the file system not found
    then
        echo "$1 is not mounted"
    fi
fi

```

output_ls.sh

```
<<DOC
Name: Babu Malagaveli
Date:27.04.2023
Description: A18 - WAS to print contents of a directory without ls command
Sample Input:
Sample Output:
DOC
#!/bin/bash
if [ $# -eq 1 ]                        #validating the output
then
    echo *                             #listing out all the exieting contents
else
    for i in $@                        #looping throught the contents
    do
        if [ -d $i ]                  #entering into the path of directory $i
        then
            cd $i                      #changing the path to home to home
            echo $i
            echo *                     #printing its contents
        else
            echo "Cannot access ‘Test’ : No such a file or directory."
        fi
    done
fi


```

user_ids.sh

```
<<DOC
Name: Babu Malagaveli
date: 29.04.2023
description: Count the number of users with user IDs between given range.
sample input:
 ./user_ids.sh 0 100
sample output:
Total count of user ID between 0 to 100 is : 3
DOC

#!/bin/bash
array=(`cat /etc/passwd | cut -d ":" -f3`)                          #cut all the IDs from the specified dir and store in array
array_count=${array[@]}                                             #counting the total no of IDs in array and storing in array_count variable
count=0                                                             #initializing the count to 0
if [ $# -eq 2 ]                                                     #if the user give the range perfectly like 100 200
then
if [ $2 -gt $1 ]                                                    #and that too if the second range is gt than first range
then
for i in $array_count                                               #looping through the all the IDs in range to compare
do
if [ $i -ge $1 -a $i -le $2 ]                                       #if the ID is in range
then
count=$((count + 1 ))                                               #inreasing the count by 1
fi
done
echo "Total count of user ID between $1 to $2 is : $count"          #when the loop ends printing the count
else
echo "Error : Invalid range. Please enter the valid range through CL."   #else if the user input is not satisfied like $2 is le $1 the error
fi
elif [ $# -eq 1 ]                                                   #if the user provides only one number as range , Then error
then
echo "Error : Please pass 2 argumrnts through CL."
echo "Usage :./user_ids.sh 100 200"
else
for i in $array_count                                               #else for the default case checking if how many IDs are b/w 500 & 10000
do
if [ $i -ge 500 -a $i -le 10000 ]
then
count=$((count + 1))                                                #increasing the count by 1 and finally printing
fi
done
echo "Total count of the user between 500 to 10000 is: $count"
fi

```

executable_path.sh

```
<<DOC
Name: Babu Malagaveli
Date:05.05.2023
Description:For each directory in the PATH, display the number of executable files in that directory
Sample Input:./executable_path.sh
Sample Output:
Current dir: /usr/local/sbin
current count: 0
Total - 2445
DOC
clear                                                               #clearing the screen for the clear output
total_x_files=0                                                     #initializing the total executable files to be zero
PATH=`echo $PATH | tr ":" " "`                                      #storing all the directories seperated by space in a variable PATH
#echo $PATH
for i in ${PATH[@]}                                                 #looping through the each directory
do
xfiles_count=0                                                      #we're here to count the number of exe files , so assuming it to be 0 initially
all_files=ls                                                        #now viewing all the files in the directory of iteration $i and storing in all_files
for j in ${all_files[@]}                                            #And looping through all the files in all_files under dir $i
do
if [ -f $j ]                                                        #if $j is a file
then
if [ -x $j ]                                                        #if $j is an exe file
then
xfiles_count=$((xfiles_count+1))                                    #then increasing the count by 1
fi
fi
cd $i                                                               #now at the end of the loop changing the dir
done
echo "Current dir: $i"                                              #printing the current dir each time for each dir
echo "current count: $xfiles_count"                                 #and the count of the exe files
total_x_files=$((xfiles_count+total_x_files))                       #as well as adding the overall executable files in each dir
done
echo "total= $total_x_files"                                        #finally printing the total number of exe files

```

replace_DEL.sh

```
<<DOC
Name: Babu Malagaveli
Date: 06.05.2023
Description: Write a script to replace 20% lines in a C file randomly and replace it with the pattern
Sample Input: ./replace_DEL.sh main.c
Sample Output:
Before replacing
#incude <stdio.h>
int main()
{
        printf("Hello world\n");
}
After replacing
#incude <stdio.h>
int main()
{
<-----------Deleted----------->
}
DOC
#!/bin/bash
no_of_lines=`cat $1 | wc -l`                                    #count the lines of the filename from the CLA
percent=$(($(($no_of_lines*20))/100))                           #calculating the 20 percent of the no of lines present
random_lines=(`shuf -i 1-$no_of_lines -n$percent`)               #getting the no of random lines which is 20% of the total lines
loop_cnt=${#random_lines[@]}                                    #let the loop run based on the no of random lines
#clear
if [ $# -eq 1 ]                                                 #checking if the user pass a file through CLA
then
    if [ -f $1 ]                                                #If the user pass a file then reading if it a file or not
    then
        if [ -s $1 ]                                            #if it is a file, checking if it contains any data
        then
            echo "Before replacing"                             #so, if it contains data , printing the data before being replaced
            cat $1
            if [ $no_of_lines -ge 5 ]                           #and going forward if and only if the content is >= 5 lines
            then
           for i in `seq 0 $(($loop_cnt-1))`                    #looping through each random line
           do
            sed -i "${random_lines[$i]}s/.*/<-----------Deleted----------->/" $1      #and relacing that random line with some content
           done
           echo "After replacing"                              #and when the loop ends, printing the change after replacement
           cat $1
            else
                echo "Error : $1 contains less than 5 lines"    #else case if the content id less than 5 lines
            fi
        else
            echo "Error : $1 is empty file. So can't replace the string."       #else case if the file is empty
        fi
    else
        echo "Error : No such a file."                          #else case if the user provides unknown file or which doesn't exist
    fi
else
echo "Error : Please pass the file name through command line."  #else case of the user didn't pass the filename thrugh CLA
fi
```