Name:Divyanshi Thapa
SAP: 590023760
SOCS Batch 17
Linux Lab Assignment 6

Submitted to: Vibhu Gautum Sir

① factorial.sh

aim: to find factorial of a number passed with the script name.

```
#!/bin/bash
# for the criteria of one integer input only.
if [ $# -ne 1 ]; then
        echo "Input one at a time".
        exit 1
fi

~~for i in "$~~
if ! [[ "$1" =~ ^[0-9]+$ ]]; then
        echo "Input a +ve no, greater than
               0".
        exit 1
fi

~~filename~~
number=$1
# a function for factorial that can be used over.
fact (){
    no="$1"
    vessel=$no.
    answer=1
    for ((i=0; i < $vessel; i++));
    do
        answer=$((answer * no))
        no=$((no -1)).
    done
    echo "$answer"
}
fact $1.
```

{Called the f^n & global first pos arg is passed

Expected Output:
./factorial.sh 4
output → 24.

→ if one no. of input isn't passed then back off

→ if the pos arg passed isn't a number then, back.
- =~ → matches if the LHS matches RHS.
- ^ → start of "$1"
- [0-9] → +ve integers.
- +$ → end till, ~~keep adding~~.

→ number to find fact is first pos arg.

→ made a f^n named fact():
Algorithm:
- → the number is the local first pos arg.
- → it is multiplied by answer variable by a for loop, in each iteration number is (-1).
- → this loop continues till "i" becomes greater the no. itself, i was 0 initially.

PAGE No.
DATE

② ~~[illegible]~~ factorial.sh
aim: to find factorial of a input number.

```
#!/bin/bash
# set the criteria for exactly one positive integer.
if [ $# -ne 1 ]; then
        ~~pif~~
        echo " Input one arg at a time".
        exit 1.
fi
if ! [[ "$1" =~ ^[0-9]+$ ]] ; then
        echo "Input ~~rt~~ +ve integer only".
        exit 1.
fi.
number= $1.              #global.
fact() {
    no.= $1              #local.
    v= $no
    ans= 1
    for ((i=0, i< $v ; i++)) ;do
            ans=$((ans * no.))
             no=$((no -1)).
    done.
    echo "$ans".

  }
fact $1                  # global first pos arg.
```

called the function with the first positional argument as input.

PAGE No.
DATE

(pos. arg).
→ If one input isn't given then back off.

→ if the pos arg ($1) doesn't have +ve integers only then back off.

→ garbage (just for my example)

→ created a fⁿ named fact():

Algorithm:
- → set the number to be equal to first arg (local).
- → ~~store it in a ~~[illegible]~~ diff variable for the loop.~~
- → now in each iteration of for loop, multiply $ans by number & subtract -1 from number.
- → this'll go on until the ~~number~~ i is less than the no. itself.

→ then call the fⁿ with global pos arg ~~[illegible]~~.

# Output:

```
./factorial.sh   3
→6
```

PAGE No.
DATE

(3) factorial.sh

```
#!/bin/bash
# criteria for single integer number input.
if [ $# -ne 1 ]; then
        echo " One input at a time".
        exit 1
fi
if ! [[ "$1" =~ ^[0-9]+$ ]]; then
        echo "~~I should~~ only input +ve
              integers".
        exit 1
fi
fact() {
    no=$1        #local .
    v=$no        # local
    a=1
    for ((i=0; i<$v; i++)); do
              a=$((a * no))
              no=$((no -1)).
    done .
    echo "$a".
}
# function for factorial.
fact $1        #global first pos arg

called function.
```

PAGE No.
DATE

→ if total / if pos arg isn't passed then back off.

→ "$1" → first pos arg .

=~ → operator that matches the arg if its digits are b/w 0 to 9

+$ → till its last digit (of $1).

→ created a f^n named 'fact'.

↓

contains a for loop that goes on until we reach the last no. of digit in the $1 #global

→ in each iteration :
var $a is mult by number & number is subt by 1.

→ at the end, we print the var $a .

then call the f^n globally .

```
# input : ./factorial.sh 2

# output : 2
```

(4) factorial.sh
AIM: Find Factorial of an input integer.

```
#!/bin/bash

if [ $# -ne 1 ]; then
        echo " Input one input at a time"
        exit 1
fi
```

if one input isn't passed, program exits.

```
if ! [[ $1 =~ [0-9]+$ ]]; then
            echo " Input a +ve integer only"
            exit 1
fi

fact() {
        n=$1        #local
        v=$n
        a=1
        for ((i=0; i<$v; i++)); do
                a=$((a*n))
                n=$((n-1))
        done
        echo "$a"
}

fact $1
```

function called.

→ shebang

→ if one inp isn't passed then back off

→ if the input (pos arg) isn't a +ve integer then back off.

→ a f^n named fact :-

Algo:-
~~answer~~ 'a' var is mult by no. till we iterate digit no. of times in each loop no.-1

end:
call f^n with global pos arg

x ——— x

input → ./factorial.sh 5

output → 120

PAGE No.
DATE

(5) factorial.sh

AIM: find the factorial of inputted number.

```
#!/bin/bash

if  [ $#  -ne  1 ]; then
        echo   " Input  1 input  at  a time"
        exit 1
fi
# now checking if the input is a valid integer.
if ! [[ $1 =~  ^[0-9]+$  ]]; then
          echo   " Inp  one +ve int only"
          exit 1
fi

fact() {
      n=$1
      r=$n
      a=1
      for  ((i=0; i<$r ; i++)); do
              a=$((a*n))
              n= $((n-1)-1))
      done
      echo  "$a"
}
# called the function.
fact $1
```

PAGE No.
DATE

→ if one input isn't passed then back off.

→ if a +ve integer isn't passed then, back off.
$1 → global first arg.
=~ [0-9] → matches the start
+$ → till end
} if its an integer (+ve)

→ creates a fⁿ
Algo:
answer var is multiplied by number till we ~~re~~ iterate digit no. of times,
In each loop no.-1

end;
call the fⁿ with global ~~fn~~ pos arg.

OUTPUT:-
input: ~~$~~ ./factorial.sh 4
output: 24

DATE

(1) #array-printing.sh .

aim : take input for array .
loop (for)
output elements.

#!/bin/bash .	-> (shebang) makes the compiler expect a bash script .
arr= ($@)	-> initialize all input (separately)
length= $# .	-> $# gives total no. of positional argument passed with script .
# running a for loop .	
for((i=0; i<"$#"; i++)); do .	
echo "${arr[$i]}"	-> prints the element at i index
done .	-> closes loop.

```
# input : ./array_printing.sh    1  4  49
#output : 1   4   49
```

⑦ #array-printing.sh
aim : take input element & print.

```
#!/bin/bash

~~arr=($@)~~
arr=($@)                         -> $@ all args as individual string.
length= $#                       -> gives the no. of positional args.

for((integer=0; i<"$#"; i++)) ;do .
        echo "${arr[$i]}".       -> prints the array element.

done .
```

```
# input : ./array-printing.sh  11 44 4844

#output : 11  44  4844
```

③ #array-printing.sh.

aim: take input elements for arr & print.

```
#!/bin/bash.                                   -> shebang for bash script.

arr=($@)                                       -> all args (positional) as individ. strg
length= ~~($@)~~ ($#)                          -> gives the total posit. args passed.

for((integer; i<"$#"; i++)); do
              echo "${arr[$i]}".               -> prints the element at i index

done.
```

```
#input: ./array-printing.sh   4   2   5

# output: 4   2   5
```

done. -> closes loop.

(4) #array-printing.sh.

aim: take ipat elements for array & print each by loop.

```
#!/bin/bash                      ->shebang.

arr=($@)                         ->all args as ind. stg.
length=$#.                       -> gives all args number.

for((i=0; i<$#; i++)); do.
        echo "${arr[$i]}".       ->prints element at i.

done
```

done .

```
# input → ./array_printing.sh  2  5  6

# output → 2  5  6
```

⑤ array-printing.sh.

aim: take input elements for array & print them.

```
#!/bin/bash.                      -> shebang.

arr=($@)                          -> all args as ind. string.
length=$#.                        -> ~~all~~ no. of total passed args.

for ((i=0; i<$#; i++)); do.
        echo "${arr[$i]}"         -> prints the. element ~~as~~ i. at i.

done.
```

input -> ./array-printing.sh 2 5 6

output -> 2 5 6

PAGE No. ____
DATE

① count_lines_words.sh.

aim : to count elements of ~~an~~ a script.
(l, w, c).

```
if  [$# -ne 1]; then.
        echo "Iput one input at a time".
        exit 1.
fi.

~~lines=$(~~
if [![-f  "$1"]]; then
        echo "Not Found".
        exit 1.
fi.

lines= $(wc -l < "$1").
word = $(wc -w < "$1")
char = $(wc -c < "$1")

echo "Lines: $lines
 Words : $word
 characters: $char".
```

PAGE No. ____
DATE

→ if one pos arg isn't passed then back off.

→ if file not fount then back.
"$1" → first pos arg.

→ uses wc-l (for no. of lines) in < "$1"
↓
filename.

→ uses wc-w (for no. of ~~lines~~ words) in < "$1"
filename)

→ uses wc-c (for total characters no.) in < "$1

→ then points everything by echo.

```
# input: ./count_lines-words.sh  script.sh.

# output: Lines: 2
          Words: 44
          character: 121
```

count - wc.sh
(c) ~~check~~ file.sh

```
#!/bin/bash.

if [ $# -ne 1 ]; then
        echo "Input one input at a time"
        exit 1.
fi.

if [ ! [ -f "$1" ] ]; then
        echo "Not found"
        exit 1.
fi.

lines = $(wc -l < "$1").
words = $(wc -w < "$1")
chars = $(wc -c < "$1").

echo "Lines : $lines
Words : $words
characters : $chars".
```

→ shebang for bash scripts.

→ If one ~~input~~ pos arg not passed then back off.

→ if file doesn't exists then print "Not found" & back off.

→ $(wc -l < "$1").
~~offers~~ returns no. of lines in the passed file. (here "$1").

→ $(wc -c < "$1") & $(wc -w < "$1") for characters & words resp.

→ print all variables using echo.

Input : ./count_lines-words.sh script.sh.

```
# output : Lines: 12
           Words: 440
           characters: 1290
```

PAGE No.
DATE

③ cnt_lwc.sh
aim: Calculate total words, lines, characters of an exist. file.

```
#!/bin/bash

if [ $# -ne 1 ]; then
        echo "Print one input at a time".
fi

if [ ! [ -f "$1" ]]; then
        echo "File not found".
fi.

lines = $(wc -l < "$1").
words = $(wc -w < "$1")
char = $(wc -c < "$1").

echo "lines: $lines
words: $words
chars: $char".
```

PAGE No.
DATE

→ for bash script compilation.

→ if total no. of pos args passed not equal to 1, then back off.

→ if file not found then back off.

→ initialize & set value of variables → line, words, chars.

used $(wc -l < "$filename") for lines

gives total lines of code in $1.

& $(wc -w < "$filename") for words.

& $(wc -c < "$filename") for characters.

then print the variables by 'echo'

input: ./cnt_lwc.sh main.txt.

```
# output: Lines: 20
          Words: 44
          Chars: 89
```

PAGE No.
DATE

(4) cnt_lwc.sh.
aim: print total words, chars & lines of an existing file.

```
#!/bin/bash.

if  [ $#  -ne 1 ]; then
        echo  "Input one input at a time"
        exit 1.

fi
if  [ ! [ -f  "$1" ] ]; then
        echo  "Not found".
        exit  1.
fi

lines = $(wc  -l  <  "$1").
w = $(wc  -w  < "$1")
c = $(wc  -c  < "$1").

echo  "Words: $w
Lines:  $lines.
chars:  $c".
```

PAGE No.
DATE

→ for bash script compilation.

→ if total 1 pos arg isn't passed then back off

→ if file not found, then say 'Not found', & back off.

→ initialize & set value of variables equal to:

$(wc -l <"$1") {lines}

$(wc -c < "$1") {chars}

$(wc -w < "$1") {words}

gives total lines of code in $1.

all for file "$1".

→ then print by echo.

input: ./cnt_lwc.sh hi.txt.

output: Words: 4
Lines: 1
Chars: 29

PAGE No. DATE

(5) cnt-lwc.sh

aim: print total words, chars & lines of an existing file.

```
#!/bin/bash

if [ $# -ne 1 ]; then
        echo "Input 1 at a time"
        exit 1
fi

if [ ! [ -f "$1" ] ]; then
        echo "Not found"
        exit 1
fi

lines=$(wc -l <"$1")
ch = $(wc -c < "$1")
w = $(wc -w < "$1")

echo "Lines: $lines
     words: $w
     characters: $ch".
```

PAGE No. DATE

→ for bash script compilation

→ if exact one no. of pos arg not passed then back off

→ if file passed not found, then back off

→ } initialize the variable names lines, ch, w by equating them to.

```
          $(wc -l < "$1")      { lines }  }
gives     $(wc -w < "$1")      { words }  } for the
no. of    $(wc -c < "$1")      { chars }  } name "$1"
lines
```

then print by 'echo.'

#input: ./cnt-lwc.sh file.c

#output: Lines: 5
Words: 24
Characters: 42

① # check-file.sh

aim: checks file & does improvisions on saying yes.

```
#!/bin/bash.

if [$#  -ne 1]; then
        exit 1
fi.

file="$1"

if [-e "$file"]; then.
    echo "file found at : $(~~$~~realpath $file)
    read -p "You wanna add anything?" input.
    If [ $ input == [Yy]* ]; then.
            cat >> $file.
    else.
        echo "Content:\n"
        cat -- "$file".
    fi.

else.
    echo "$file doesn't exist" ~~You wann~~
    read -p "You wanna create one?" ans.
    if [ $ans == [Yy]* ]; then.
            touch $file.
            echo "$file created".
    else.
        echo "Not creating one".
        exit 1
    fi.
fi.
```

→ she bang.

→ no. of total pos args passed ne to 1, then back off.

→ filename is first pos arg.

→ If file exists then:

show its path.
show content. (use cat).

if not;
- ask to create one instead.
- If wants to:
  - use 'touch'.
- else:
  - print okay.

#input: ./check-file.sh existing-file.c

#output:

```
file found at : /home/vboxuser/Docs/existing-file.c.
You wanna add anything?
y
Hi.
```

PAGE No.
DATE

② # check-file sh.
aim: to search a file, if not found then create one instead.

```
#! /bin/bash

if [ #$ -ne 1 ]; then
        echo "Input one arg at a time".
        exit
fi.

file="$1"

if [-e $ file]; then
        echo "file found at $ (realpath $ file)"
        read -p "You wanna add anth?" ans.
        if [ $ ans == [Yy]* ]; then
                cat >> $file.
        else.
        ~~con~~ echo "content:\n".
        cat -- "$ file".
else.
    read ~~*~~p "You wanna create one instead?" inp.
    if [$inp == [Yy]* ; then.
            touch $ file.
            echo "file created".
    else.
        "Okay".
fi.
```

PAGE No.
DATE

→ shebang.

→ if exactly one ipu isn't passed then back off

→ first arg stored as ~~firs~~ filename.

→ if file exits then;.
- show path.
- ask if it wants to create one more as well.
- if yes;
  use cat,
- else;
  just show content.

else;
  ask to create one instead.
  If yes;
    use touch.
  else;
    "Okay".

input: ./check-file.sh non-exist.txt

output:
You wanna create one instead?
N.

PAGE No.
DATE

③ # check-file.sh

```
#!/bin/bash

if [ $# -ne 1 ]; then.
        echo " Iput 1 ~~ip~~ input at a time".
        exit 1
fi.

file="$1"

if [-e $file]; then.
        echo "Path: $(realpath $file)".
      read -p "Wanna add content ?:" inp.
      ~~read~~
      if [$inp == [Yy]*]; then.
            ~~ed~~ cat >> $file.
      else.
        echo"content :".
          cat -- $file.

else.
    ~~echo~~
    read -p "Wanna create one instead ?" ans.
    if [ $ans == [Yy]*]; then.
            touch $file.
            echo "Created".
    else
      echo"Okay".

fi.
```

PAGE No.
DATE

→ the bang.

→ stores first pos arg as filename.

→ if no. of arg isn't equal to 1, then back off

~~fi~~.

→ if file exists then:

```
        show path.
        show content.
        ask to add anything:
            if yes:
                use cat >>.
            else:
                back off.

else.
    ask to create one instead;
    if yes:
        use touch.
    else:
        echo "Okay".
```

input: ./check-file.sh main.sh

#output: Wanna create one insted?
Y
"Created".

shebang. →

if no. of args passed → isn't 1 then back off.

store filename as → first arg passed.

if file exists then do → this:

- if yes;
  - (i) show ~~conte~~ path.
  - (ii) Ask if it wants to add anything.
  - (iii) if yes then use cat.
- else. just show the content.

else.
create one..
if yes: (used [Yy]*).
create it.
else:
don't.

input: ./check-file.sh quiet.c hi.c

output: Input 1 input at a time.

④ #check-file.sh

```
#!/bin/bash.

file=$1.

if [ $# -ne 1]; then
        echo "Input 1 input at a time".
        exit 1.
fi.

if [-e $file] ~~-ne~~; then.

        echo "Found at : $(realpath $file)".
        echo "content:\n".
        cat -- $file.
        read -p "Wanna add anything ?" ans
        if [ $ans == [Yy]* ]; then.
                cat >> $file.
        else.
                exit 1.

else.
        read -p "Wanna create one instead" ~~ans~~ answ
        if [ $answ == [Yy]* ]; then.
                ~~cat >> $file~~.
                touch $file.
                echo "created"
        else.
                echo "okay".

fi.
```

⑤ check_file.sh

```
#!/bin/bash

if [ $#  -ne 1 ]; then
        echo "Iput 1 input at a time".
        exit 1.

fi.

file="$1".

if  [-e  $file] ; then
        echo  "file found: $(realpath $file)"
        ~~read~~
        echo  "Content : "
        cat  -- $file.

else.
    read -p "Wanna create one instead ?" ans
    if  [$ ans == [Yy]*] ; then
            ~~echo~~
            touch  $file.
            echo  "created".

    else.
        echo "Okay".

fi.
```

→ the shebang.

→ if no. of arg passed isn't equal to 1, then return back

→ filename to find is first argument.

→ if file exists :
  show path.
  ask if user wants to add content.
  if yes :
      use cat for that.
  else :
      just show the content.

else :
  ask create one.
  if yes :
      Use touch for that.

  else ;
      "okay".

```
#input : ./check_file.sh  can't_find.t

#output : Wanna create one instead?
          Y
          created
```