1. Check file

Problem Statement :

Write a shell script that checks whether a given file exists or not.
If the file exists, it displays its contents.
If the file doesn't exist, it asks the user whether they want to create it.

Expected Output

Case 1 : File Exists

```
$ ./check-file.sh notes.txt
File exists : notes.txt
---- contents ----
This is my note file.
```

Case 2 : File does not exist

```
$ ./check_file.sh newfile.txt
File 'newfile.txt' does not exist.
Create it now ? (y/N) : y
Created newfile.txt
You can edit it using your favourite editor.
```

1. Check file

```
#!/bin/bash
# check_file.sh
# Usage: ./checkfile.sh filename.txt
```

} Comment: shows how to execute / run the script

```
if [ $# -ne 1 ] then
  echo "Usage: $0 <filename>"
  exit 1
fi
```

: $# = no. of arguements, -ne 1 = not equal to 1

. If the user does not provid exactly one arguement, it prints the usage message and exits with status 1.

```
file = "$1"
if [ -e "$file" ]; then
  echo "File exists: $file"
  echo "---- contents ----"
  cat -- "$file"
else
  echo "File '$file' does not exist"
  read -p "Create it now? [y/N]: " ans
  case "$ans" in
  [Yy]*) touch "$file"; echo "Created $file";
  echo "You can edit it using your favourite editor";;
  *) echo "Not creating file";;
  esac
fi
```

. stores the first arguement into a variab (file)

. checks if the file exists

. displays the file's content

. runs when file does not exist

. read -p = asks the to create a file

. statement

. creates a fi

2. Check file

```
#!/bin/bash
# checkfile.sh
# Usage: ./check_file.sh filename.txt

if [$# -ne 1] then
   echo "Usage: $0 <filename>"
   exit 1
fi

file = "$1"
if [-c ""$1file"]; then
   echo "File exists: $file"
   echo "---- contents----"
   cat -- "$file"
else
   echo "File '$file' does not exists"
   read-p "Create it now? (y/N): "ans
   case "$ans" in
    [Yy]*) touch "$file"; echo "Created'$file";
   echo "You can edit it using your favourite editor" ;;
   *) echo "Not creating file." ;;
   esac
fi
```

__/__/____

3. Check_file.sh

```
#!/bin/bash
#check_file.sh
#Usage: ./check_file.sh filename.txt

if [$# -ne 1]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

file = "$1"
if [-e "$file"]; then
  echo "File exists: $file"
  echo "---- contents ---"
  cat -- "$file"
else
  echo "File '$file' does not exist."
  read -p "Create it now? (y/N): " ans
  case "$ans" in
  [Yy]*) touch "$file"; echo "Created $file;
  echo "You can edit it using your favourite editor.";;
  *) echo "Not creating file.";;
  esac
fi
```

- Tells the OS to run the script using bash
- Comments: describes what the script does & how to use it

4. Check file

```
#!/bin/bash
# check-file.sh
# Usage : ./check-file.sh filename.txt

if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

file = "$1"
if [-e "$file"] ; then
  echo "File exists : $file"
  echo "--- contents ---"
  cat -- "$file"
else
  echo "File '$file' does not exist"
  read -p "Create it now? (y/N):" ans
  case "$ans" in
  [Yy]*) touch "$file" ; echo "Created $file";
  echo "You can edit it using your favourite editor"
  *) echo "not creating file" ;;
  esac
fi
```

5. Check file

```
#!/bin/bash
# check_file.sh
# Usage: ./check_file.sh filename.txt

if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

file = "$1"
if [-e "$file" ]; then
   echo "File exists: $file"
   echo "--- contents ---"
   cat -- "$file"
else
   echo "File '$file' does not exist"
   read -p "Create it now? (y/N):" ans
   case "$ans" in
   [Yy]*) touch "$file"; echo "Created $file";
   echo "You can edit it using your favourite ed
   *) echo "Not creating file";;
   esac
fi
```

__/__/____

2. Count Lines, Words & Characters

Problem Statement

Write a shell script that counts the number of lines, words and characters in a given text file.

Example Output:

```
$ ./count_lwc.sh sample.txt
Lines : 5
Words : 20
Characters : 110
```

1. Count Lines, Words & Chars

```
#!/bin/bash
```
. Tells OS to run the script using bash

```
# count_lwc.sh
# Usage: ./count_lwc.sh filename.txt
```
} Comment: shows how to run the script.

```
if [ $# -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi
```
. Checks if no. of arguements ($#) is not equal to 1 (-ne 1)
. prints a usage message
. exits the script with status 1

```
if [ ! -f "$1" ]; then
    echo "File not found"
    exit 1
fi
```
. checks whether the path exists or not.
. prints an error message
. exits the script with status 1.

```
lines = $(wc -l < "$1")
words = $(wc -w < "$1")
chars = $(wc -m < "$1")
```
. counts the no. of lines & stores it in 'lines'.
. counts the no. of words & stores it in 'words'
. counts the no. of characters & stores it in 'chars'.

Prints the -

```
echo "Lines: $lines"
echo "Words: $words"
echo "Chars: $chars"
```
. line count
. word count
. character count.

2. Count Lines, Words & Chars

```
#!/bin/bash
# count-lwc.sh
# Usage: ./count-lwc.sh filename.txt

if [$# -ne 1]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

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

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

echo "Lines: $lines"
echo "Words: $words"
echo "Chars: $chars"
```

3. Count Lines, Words & Chars

```
#!/bin/bash
# count-lw.sh
# Usage: ./count-lw.sh filename.txt

if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

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

lines = $(wc -l < $1)
words = $(wc -w < $1)
chars = $(wc -m < $1)

echo "Lines : $lines"
echo "Words : $ words"
echo "Chars : $ chars"
```

4. Count Lines, Words & Chars

```
#!/bin/bash
# count_lw.sh
# Usage: ./count-lw.sh filename.txt

if [$# -ne 1]; then
   echo "Usage: $0< filename>"
   exit 1
fi

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

  lines = $(wc-l <$1)
  words = $(wc-w<$1)
  chars = $(wc-m<$1)

echo "Lines: $lines"
echo "Words: $words"
echo "Chars: $chars"
```

5. Count Lines, Words & Chars

```
#!/bin/bash
# count-lwc.sh
# Usage: ./count-lwc.sh filename.txt

if [$# -ne 1]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

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

lines = $(wc -l < $1)
words = $(wc -w < $1)
chars = $(wc -m < $1)

echo "Lines : $lines"
echo "Words: $words"
echo "Chars : $chars"
```

3. Factorial

Problem Statement :

Write a shell script to calculate the factorial of one or more given non-negative integers using a function.

Example Output :

Case 1 : Single Input

```
$ ./factorial.sh 5
5! = 120
```

Case 2 : Multiple Input

```
$ ./factorial.sh 3 6 0
3! = 6
6! = 720
0! = 1
```

Case 3 : Invalid Input

```
$ ./factorial.sh 7 hello
7! = 5040
hello: not a non-negative integer, skipping.
```

1. Factorial Function

```
#!/bin/bash
# factorial.sh
# Usage: ./factorial.sh 5

fact() {
  n=$1
  if [ "$n" -le 1 ]; then
    echo 1
    return
  fi
  res=1
  for ((i=2; i<=n; i++)); do
    res=$((res*i))
  done
  echo "$res"
}

if [ $# -lt 1 ]; then
  echo "Usage: $0 <non-negative integer> [another...]"
  exit 1
fi
```

- Tells OS to execute script using bash
- Comment- shows how to run the script.
- fact() = function that calculates the factorial of a number.
- assigns 1st arguement to the variable 'n'
- initialising a variable
- checks if no arguements were pas

```
for arg in "$@"; do
  if ![[ $arg =~ ^[0-9]+$ ]]; then
  echo "arg: not a non-negative integer, skipping."
  continue
fi
  echo "$arg! = $(fact "$arg")".
done
```

2. Factorial Function

```
#!/bin/bash
# factorial.sh
# Usage: ./factorial.sh 5

fact(){
  n=$1
if ["$n" -le 1]; then
  echo 1
  return
fi
  res=1
for((i=2; i<=n; i++)); do
  res=$((res*1))
done
echo "$res"
}
```

```
if [$# -lt 1]; then
  echo "Usage: $0 <non-negative-integer> [another ...]"
  exit 1
fi

for arg in "$@"; do
  if ![[$arg =~ ^[0-9]+$]]; then
  echo "arg: not a non-negative integer, skipping"
  continue
  fi
  echo "$arg ! = $(fact "$arg")"
done.
```

3. Factorial Function

```
#!/bin/bash
# factorial.sh
# Usage: ./factorial.sh 5

fact() {
  n = $1
  if [ "$n" -le 1]; then
  echo
  return
fi
  res = 1
```

```
    for ((i=2; i<=n; i++)); do
        res=$((res * i))
    done
    echo "$res"
}

if [ $# -lt 1 ]; then
    echo "Usage: $0 <non-negative-integer> [another...]"
    exit 1
fi

for arg in "$@"; do
    if ! [[ $arg =~ ^[0-9]+$ ]]; then
        echo "$arg: not a non-negative integer, skipping."
        continue
    fi
    echo "$arg! = $(fact "$arg")"
done
```

4. Factorial Function

```
#!/bin/bash
# factorial.sh
# Usage: ./factorial.sh 5
```

```
fact() {
  n = $1
  if [ "$n" -le 1 ]; then
   echo 1
   return
  fi
  res = 1
  for (( i = 2; i <= n; i++ )); do
   res = $(( res * i ))
  done
  echo "$res"
}

if [ $# -lt 1 ]; then
  echo "Usage: $0 <non-negative-integer> [another...]"
  exit 1
fi

for arg in "$@"; do
  if ! [[ $arg =~ ^[0-9]+$ ]]; then
   echo "$arg: not a non-negative integer, skipping"
continue
  fi
  echo "$arg! = $(fact "$arg")"
done
```

4. Print Number

Problem Statement:

Write a shell script to store numbers in an array and display each element one by one using a for loop.

Example Output:

```
$ ./one_to_ten.sh
1
2
3
4
5
6
7
8
9
10
```

1. Print Number

```
#!/bin/bash
#Usage: ./one_to_ten.sh

a = (1 2 3 4 5 6 7)

for i in "${a[@]}"; do
  echo "$i"
done
```

- Tells OS to execute script using bash
- comment - shows how to run the script.
- creates an array named 'a' with values 1 2 3 4 5 6 7
- for = loop, "${a[@]}" = expands to all elements.
- prints the current value of i
- end of for loop

2. Print Number

```
#!/bin/bash
#Usage: ./one_to_ten.sh

a = (1 2 3 4 5 6 7)

for i in "${a[@]}"; do
  echo "$i"
done
```

- Tells OS to execute script using bash
- Comment - shows how do run the script
- Creates an array named 'a' with values 1 2 3 4 5 6 7
- "${a[@]}" = expands to all elemen each element is stored in i one by
- prints the current value of i
- end of for loop.

3. Print Number

```
#!/bin/bash
#Usage: ./one-to-ten.sh

a = (1 2 3 4 5 6 7)

for i in "${a[@]}"; do
  echo "$i"
done
```

- Tells OS to execute script using bash
- Comment - shows how to run the script
- creates an array named 'a' with values 1 2 3 4 5 6 7
- "${a[@]}" = expands to all elements → each element is stored in i one by one
- prints current value of i
- end of for loop

4. Print Number

```
#!/bin/bash
#Usage: ./one_to_ten.sh

a = (1 2 3 4 5 6 7)

for i in "${a[@]}"; do
  echo "$i"
done
```

- Tells OS to execute script using bash
- Comment - shows how to run the script
- creates an array named 'a' with values 1 2 3 4 5 6 7
- "${a[@]}" = expands to all elements → each element is stored in i one by one
- prints current value of i
- end of for loop.

5. Print Number

```
#!/bin/bash
# Usage: ./one-to-ten
```

. . Tells OS to run script using bash
. . Comment - shows how to run the script

```
a=(1 2 3 4 5 6 7)
```

. creates an array named 'a' with values 1 2 3 4 5 6 7

```
for i in "${a[@]}"; do
```

. "${a[@]}" = expands to all elements → each value is stored in i one by one

```
  echo "$i"
```

. . prints the current value of i

```
done
```

. . end of for loop.