Question no.1

Write a shell script program that takes the name of a file or directory as an argument.

⇨ This script first checks if an argument was provided, and if not, it displays an error message and exits. Then, it checks if the argument is a file or directory using the -f and -d options of the test command. If it's a file, it counts the number of lines in the file using the wc command, and if it's a directory, it lists the files in the directory using the ls command.

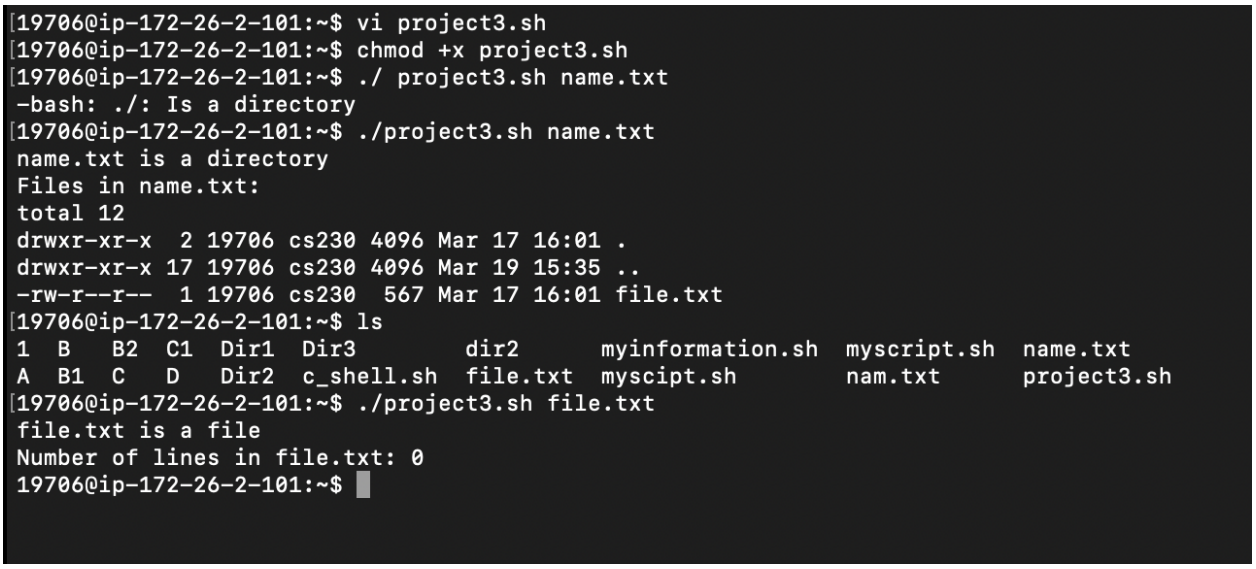
```
#!/bin/bash

# check if an argument was provided
if [ -z "$1" ]; then
    echo "Error: no file or directory provided"
    exit 1
fi

# check if the argument is a file or directory
if [ -f "$1" ]; then
    echo "$1 is a file"
elif [ -d "$1" ]; then
    echo "$1 is a directory"
else
    echo "$1 is not a file or directory"
fi

# count the number of lines in the file (if it's a file)
if [ -f "$1" ]; then
    num_lines=$(wc -l < "$1")
    echo "Number of lines in $1: $num_lines"
fi

# list the files in the directory (if it's a directory)
if [ -d "$1" ]; then
    echo "Files in $1:"
    ls -al "$1"
fi
```



- It then reports if the file is a directory, regular file or symbolic link.

⇨ In this script -d option checks if the given file path is a directory, the -f option checks if it's a regular file, and the -L option checks if it's a symbolic link. If none of these conditions are met, then it reports that the provided file is not a file or directory.

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Error: no file or directory provided"
    exit 1
fi

file=$1

if [ -d "$file" ]; then
    echo "$file is a directory"
elif [ -f "$file" ]; then
    echo "$file is a regular file"
elif [ -L "$file" ]; then
    echo "$file is a symbolic link"
else
    echo "$file is not a file or directory"
fi
~
~
~
```

```
[19706@ip-172-26-2-101:~$ vi project3.sh
[19706@ip-172-26-2-101:~$ chmod +x project3.sh
[19706@ip-172-26-2-101:~$ ls
1  B   B2  C1  Dir1  Dir3       dir2      myinformation.sh  myscipt.sh   nam.txt   project3.sh
A  B1  C   D   Dir2  c_shell.sh  file.txt  mylink            myscript.sh  name.txt
[19706@ip-172-26-2-101:~$ ./project3.sh A
A is a directory
[19706@ip-172-26-2-101:~$ ./project3.sh file.txt
file.txt is a regular file
[19706@ip-172-26-2-101:~$ ./project3.sh mylink
mylink is a regular file
19706@ip-172-26-2-101:~$
```

- It also reports if the user has read, write, and execute permission on the file or directory.

⇨ This script will first check if a file or directory is provided as an argument. If not, it will print an error message and exit. Then, it will check if the file or directory exists. If not, it will print an error message and exit.

```bash
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Error: no file or directory provided"
    exit 1
fi

FILE_OR_DIR=$1

if [ ! -e "$FILE_OR_DIR" ]; then
    echo "$FILE_OR_DIR does not exist"
    exit 1
fi

if [ -d "$FILE_OR_DIR" ]; then
    TYPE="directory"
elif [ -f "$FILE_OR_DIR" ]; then
    TYPE="regular file"
elif [ -L "$FILE_OR_DIR" ]; then
    TYPE="symbolic link"
else
    TYPE="unknown"
fi

echo "$FILE_OR_DIR is a $TYPE"

if [ -r "$FILE_OR_DIR" ]; then
    echo "You have read permission on $FILE_OR_DIR"
fi

if [ -w "$FILE_OR_DIR" ]; then
    echo "You have write permission on $FILE_OR_DIR"
fi

if [ -x "$FILE_OR_DIR" ]; then
    echo "You have execute permission on $FILE_OR_DIR"
fi
```

```
19706@ip-172-26-2-101:~$ vi project3.sh
19706@ip-172-26-2-101:~$ chmod +x project3.sh
19706@ip-172-26-2-101:~$ ./project3.sh name.txt
name.txt is a directory
You have read permission on name.txt
You have write permission on name.txt
You have execute permission on name.txt
19706@ip-172-26-2-101:~$ ./project3.sh mylink
mylink is a regular file
You have read permission on mylink
19706@ip-172-26-2-101:~$
```

- If the file/directory (the argument) does not exist, it should issue a message and exit gracefully.

⇨ A script that checks if the argument is a file or directory and prints a message accordingly. If no

argument is provided, it will print a message saying there is no file or directory available.

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: no file or directory provided"
    exit 1
fi

FILE_OR_DIR=$1

if [ -f "$FILE_OR_DIR" ]; then
    echo "$FILE_OR_DIR is a file"
elif [ -d "$FILE_OR_DIR" ]; then
    echo "$FILE_OR_DIR is a directory"
else
    echo "$FILE_OR_DIR is not a file or directory"
fi
~
~
```

```
[19706@ip-172-26-2-101:~$ vi project3.sh
[19706@ip-172-26-2-101:~$ chmod +x project3.sh
[19706@ip-172-26-2-101:~$ ./project3.sh file
file is not a file or directory
19706@ip-172-26-2-101:~$
```

Question no.2

Write a shell script program that takes two integer arguments. The second argument is assumed to be greater than the first. The output of the program is a counting of numbers beginning at the first number and ending with the second number.

⇨ The script checks whether it has received two arguments. If not, it prints a usage message and exits with an error status. Then it stores the first argument as the start variable and the second argument as the end variable. After checking whether the start is less than the end. If not, it prints an error message and exits with an error status. The script uses a for loop to count from start to end, printing each number on a new line.

```
#!/bin/bash

if [ $# -ne 2 ]; then
  echo "Usage: $0 start end"
  exit 1
fi

start=$1
end=$2

if [ $start -ge $end ]; then
  echo "Error: start ($start) must be less than end ($end)"
  exit 1
fi

for (( i=$start; i<=$end; i++ )); do
  echo $i
done
```

```
19706@ip-172-26-2-101:~$ vi project3.sh
19706@ip-172-26-2-101:~$ ./project3.sh 1 5
1
2
3
4
5
19706@ip-172-26-2-101:~$ ./project3.sh 6  5
Error: start (6) must be less than end (5)
19706@ip-172-26-2-101:~$
```

- If the second argument is less than or equal to the first, it should issue a message and exit gracefully.

⇨ This script takes two integer arguments and checks if they are both provided. If the second argument is less than or equal to the first argument, it will output an error message and exit. Otherwise, it will loop through all the integers between the first and second argument and output each number on a new line.

```
#!/bin/bash

if [ $# -ne 2 ]; then
    echo "Error: missing arguments"
    exit 1
fi

if [ $2 -le $1 ]; then
    echo "Error: second argument must be greater than first argument"
    exit 1
fi

for (( i=$1; i<=$2; i++ ))
do
    echo $i
done
~
~
~
~
~
```

```
19706@ip-172-26-2-101:~$ vi project3.sh
19706@ip-172-26-2-101:~$ ./project3.sh 1 5
1
2
3
4
5
19706@ip-172-26-2-101:~$ ./project3.sh 6  5
Error: second argument must be greater than first argument
19706@ip-172-26-2-101:~$
```

Question no.3

Write a shell script program that reads a list of numbers from standard input; the output is the sum of the numbers.

⇨ The program reads each line of input using the read command within a while loop and adds each number to a running total sum. Once all the input is read, the program outputs the final sum using the echo command.

To run the program, we can provide a list of numbers as input by typing them in manually followed by Ctrl-D to signal the end of the input.

```
#!/bin/bash

sum=0
while read -r number; do
  sum=$((sum+number))
done

echo "The sum of the numbers is: $sum"
```

```
[19706@ip-172-26-2-101:~$ vi project3.sh
[19706@ip-172-26-2-101:~$ chmod +x project3.sh
[19706@ip-172-26-2-101:~$ ./project3.sh
 1
 2
 3
 4
 5
[20
 The sum of the numbers is: 35
 19706@ip-172-26-2-101:~$
```

Question no.4

Write a shell script that will rename all the files in a directory. It will mostly be useful to rename digital picture files. The first argument is a base name, second argument is a file extension. If it is run as:

$ renumber 1stBirtday jpg
then the resulting files should have names like:

1stBirthday001.jpg, 1stBirthday002.jpg, 1stBirthday003.jpg, etc.

- ⇨ The shell script program for renaming files in a directory works by first checking if the correct number of arguments has been passed to it. If not, it displays a usage message. If the arguments are

correct, it uses a for loop to loop through all the files in the directory with the specified extension. It then renames each file with the specified base name and a sequential number. The number is padded with leading zeros to ensure that all the filenames have the same length. The script uses the "mv" command to rename the files. Finally, it displays a message indicating that all files have been renamed.

```
#!/bin/bash

# Check if the script has been given the right number of arguments
if [ $# -ne 2 ]
then
    echo "Usage: $0 <base name> <file extension>"
    exit 1
fi

# Store the base name and file extension in variables
base_name=$1
file_extension=$2

# Check if any files exist with the given file extension
if ! ls *.$file_extension > /dev/null 2>&1
then
    echo "No files found with extension .$file_extension"
    exit 1
fi

# Loop through all files with the given file extension
count=1
for file in *.$file_extension
do
    # Generate the new file name
    new_file_name=$(printf "%s%03d.%s" "$base_name" "$count" "$file_extension")

    # Rename the file
    mv "$file" "$new_file_name"

    # Increment the count
    count=$((count+1))
done

echo "All files renamed with base name '$base_name' and file extension '$file_extension'"
~
```

```
19706@ip-172-26-2-101:~$ vi renumber.sh
19706@ip-172-26-2-101:~$ chmod +x renumber.sh
19706@ip-172-26-2-101:~$ ./renumber.sh 1stBirthday jpg
mv: '1stBirthday001.jpg' and '1stBirthday001.jpg' are the same file
All files renamed with base name '1stBirthday' and file extension 'jpg'
```