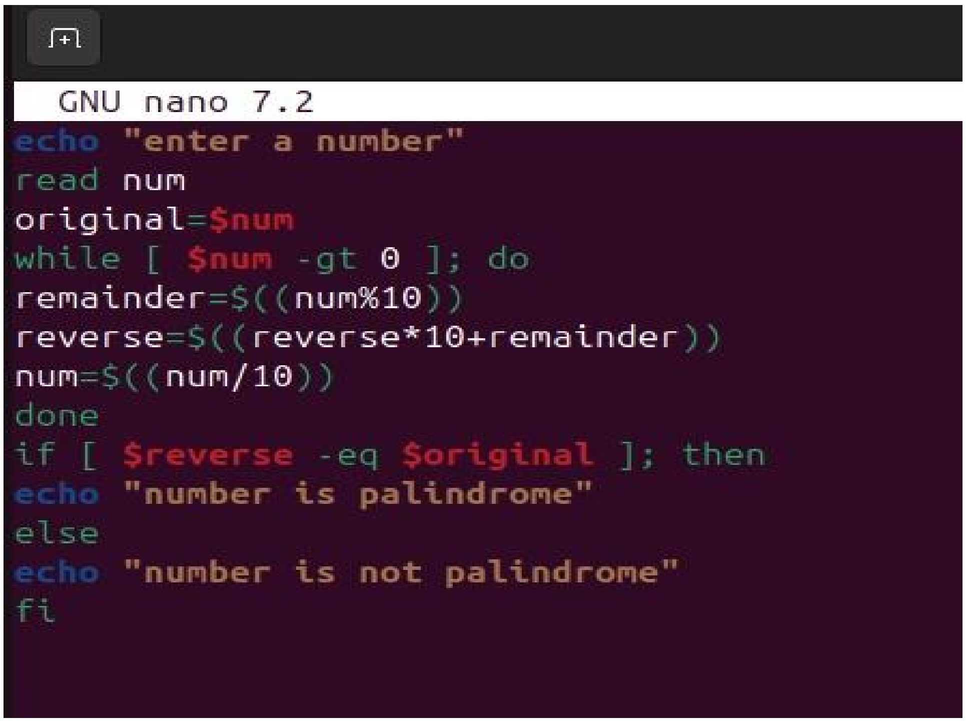
Experiment 6: Shell Programming

i. Write a script that checks whether a given number is a palindrome or not. A palindrome number reads the same backward as forward. \

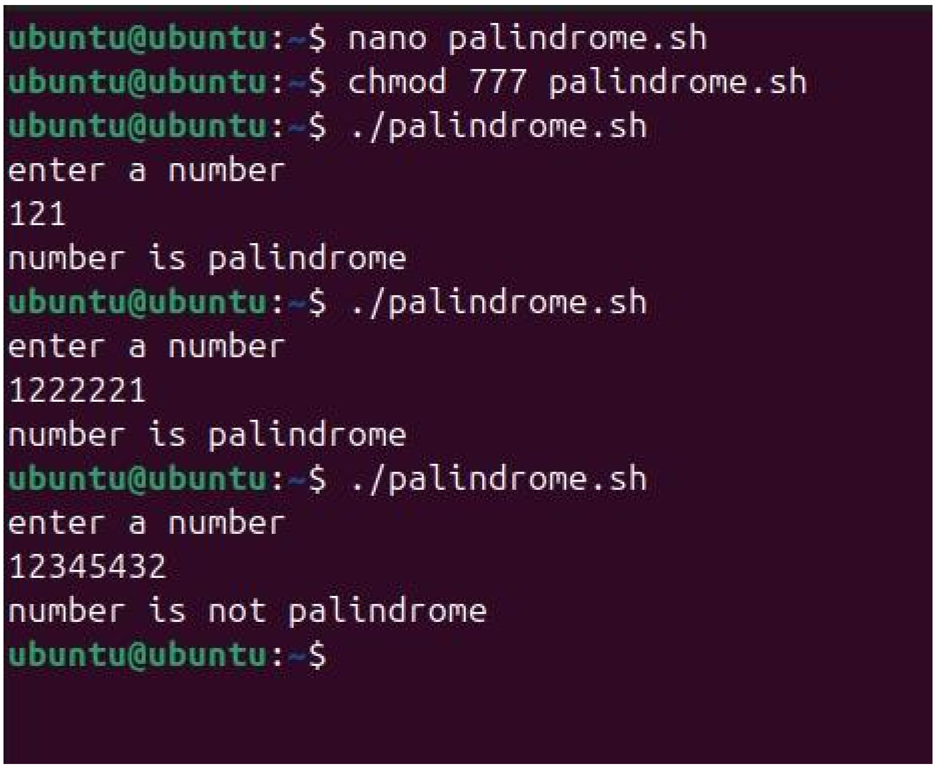
SOLUTION

- A **palindrome number** reads the same backward as forward (e.g., 121, 1331).
- Accept the number as input from the user.
- Convert the number to a string to simplify reversal and comparison.
- Reverse the string using slicing or a built-in method.
- Compare the original string with the reversed string.
- If both strings are equal, the number is a palindrome.
- If they are not equal, the number is not a palindrome.
- Display the result to the user accordingly.



```
  GNU nano 7.2
echo "enter a number"
read num
original=$num
while [ $num -gt 0 ]; do
remainder=$((num%10))
reverse=$((reverse*10+remainder))
num=$((num/10))
done
if [ $reverse -eq $original ]; then
echo "number is palindrome"
else
echo "number is not palindrome"
fi
```

Output:



```
ubuntu@ubuntu:~$ nano palindrome.sh
ubuntu@ubuntu:~$ chmod 777 palindrome.sh
ubuntu@ubuntu:~$ ./palindrome.sh
enter a number
121
number is palindrome
ubuntu@ubuntu:~$ ./palindrome.sh
enter a number
1222221
number is palindrome
ubuntu@ubuntu:~$ ./palindrome.sh
enter a number
12345432
number is not palindrome
ubuntu@ubuntu:~$
```

ii. Write a script that calculates the greatest common divisor (GCD) and the least common multiple (LCM) of two given numbers.

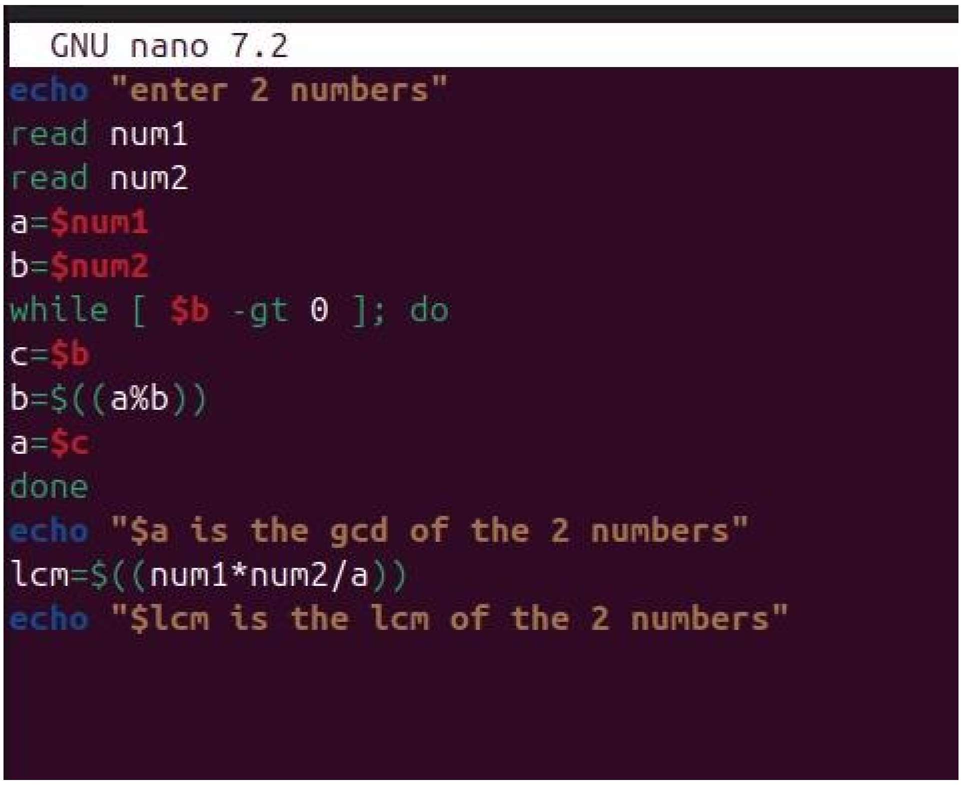
SOLUTION:

GCD (Greatest Common Divisor)

- GCD is the largest number that divides both numbers without leaving a remainder.
- Accept two numbers as input from the user.
- Use the **Euclidean algorithm** to find the GCD:
    - While the second number is not zero, replace the first number with the second, and the second with the remainder of the division of the two.
    - When the second number becomes zero, the first number is the GCD.

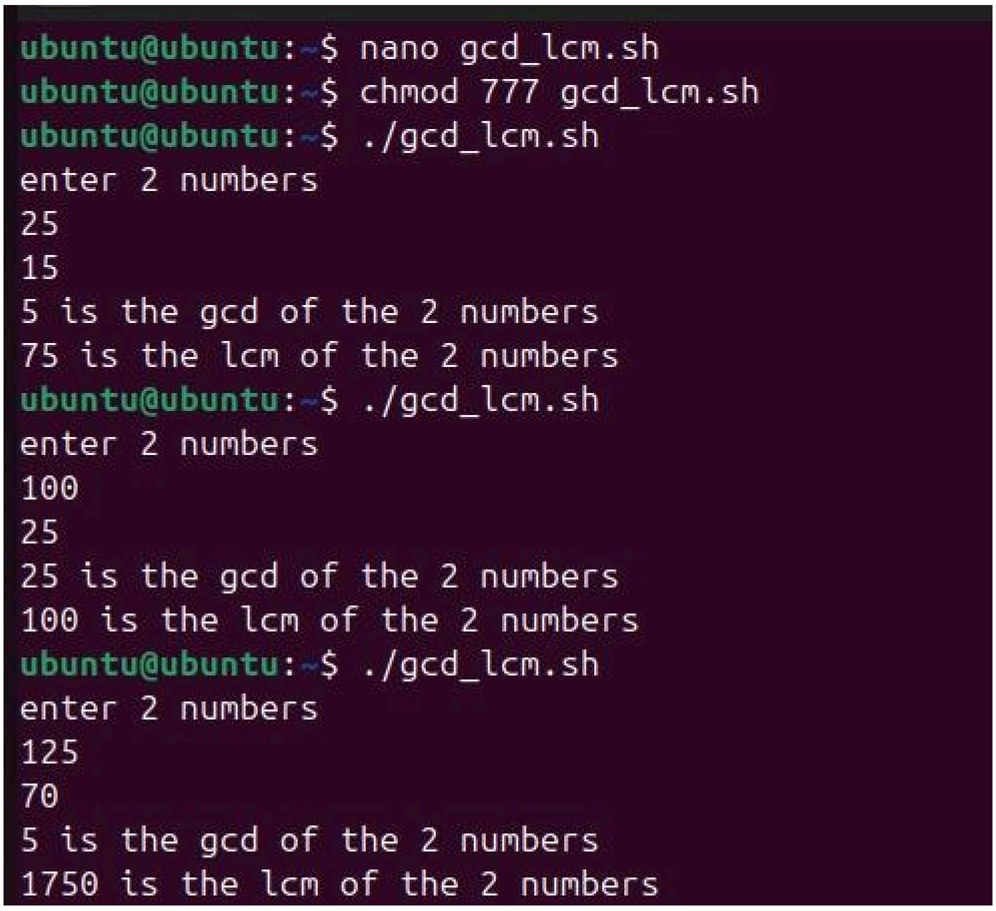
LCM (Least Common Multiple)

- LCM is the smallest number that is a multiple of both numbers.
- Use the relationship between GCD and LCM:
    - `LCM(a, b) = (a × b) / GCD(a, b)`
- Once you calculate the GCD, plug it into the formula to get the LCM.
- Display both the GCD and LCM to the user.



```
  GNU nano 7.2
echo "enter 2 numbers"
read num1
read num2
a=$num1
b=$num2
while [ $b -gt 0 ]; do
c=$b
b=$((a%b))
a=$c
done
echo "$a is the gcd of the 2 numbers"
lcm=$((num1*num2/a))
echo "$lcm is the lcm of the 2 numbers"
```

OUTPUT:



```
ubuntu@ubuntu:~$ nano gcd_lcm.sh
ubuntu@ubuntu:~$ chmod 777 gcd_lcm.sh
ubuntu@ubuntu:~$ ./gcd_lcm.sh
enter 2 numbers
25
15
5 is the gcd of the 2 numbers
75 is the lcm of the 2 numbers
ubuntu@ubuntu:~$ ./gcd_lcm.sh
enter 2 numbers
100
25
25 is the gcd of the 2 numbers
100 is the lcm of the 2 numbers
ubuntu@ubuntu:~$ ./gcd_lcm.sh
enter 2 numbers
125
70
5 is the gcd of the 2 numbers
1750 is the lcm of the 2 numbers
```

iii. Create a script that takes multiple numbers as input and sorts them in ascending or descending order.

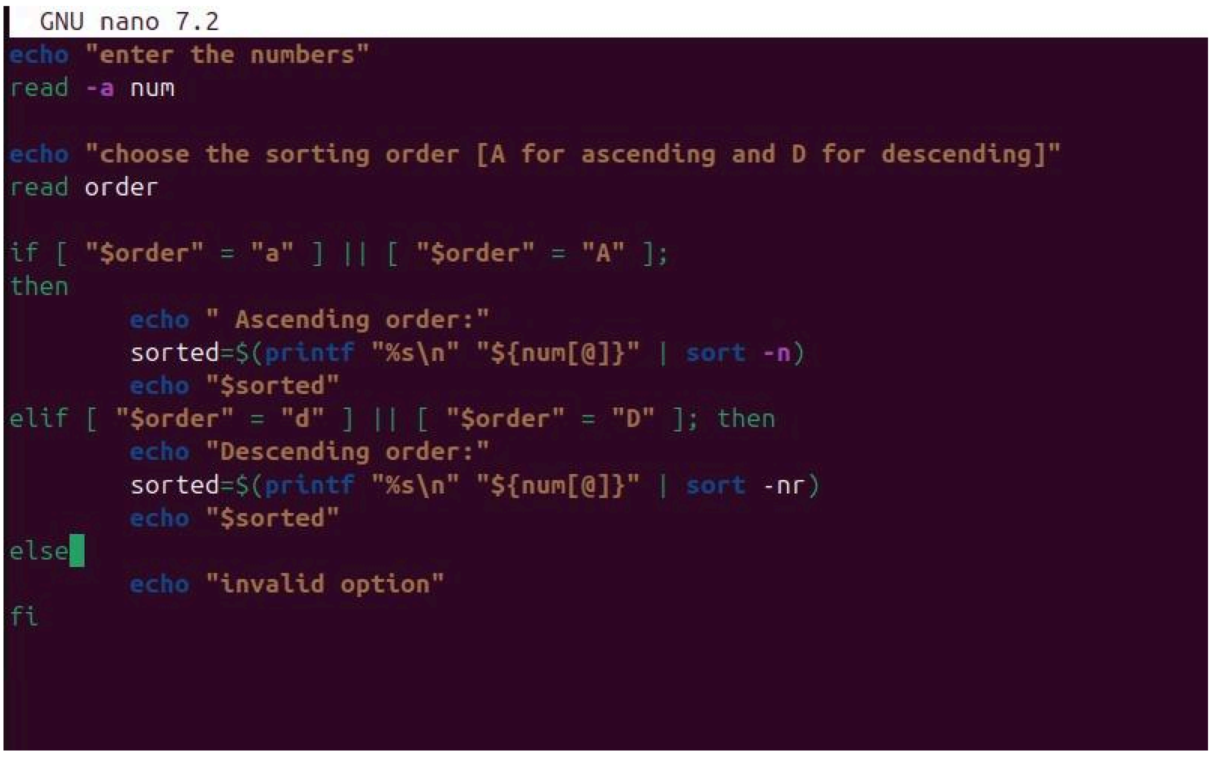
SOLUTION:

Input Handling

- Accept multiple numbers from the user.
- Input can be taken as a single line (e.g., space-separated or comma-separated).
- Split the input string into individual elements.
- Convert each element to an integer.

Sorting Logic

- Ask the user for the preferred sort order (ascending or descending).
- Use a sorting function (like `sort()` or `sorted()` in Python).
- For:
  - **Ascending order** → sort normally.
  - **Descending order** → sort in reverse.
- Display the sorted list to the user in the chosen order.

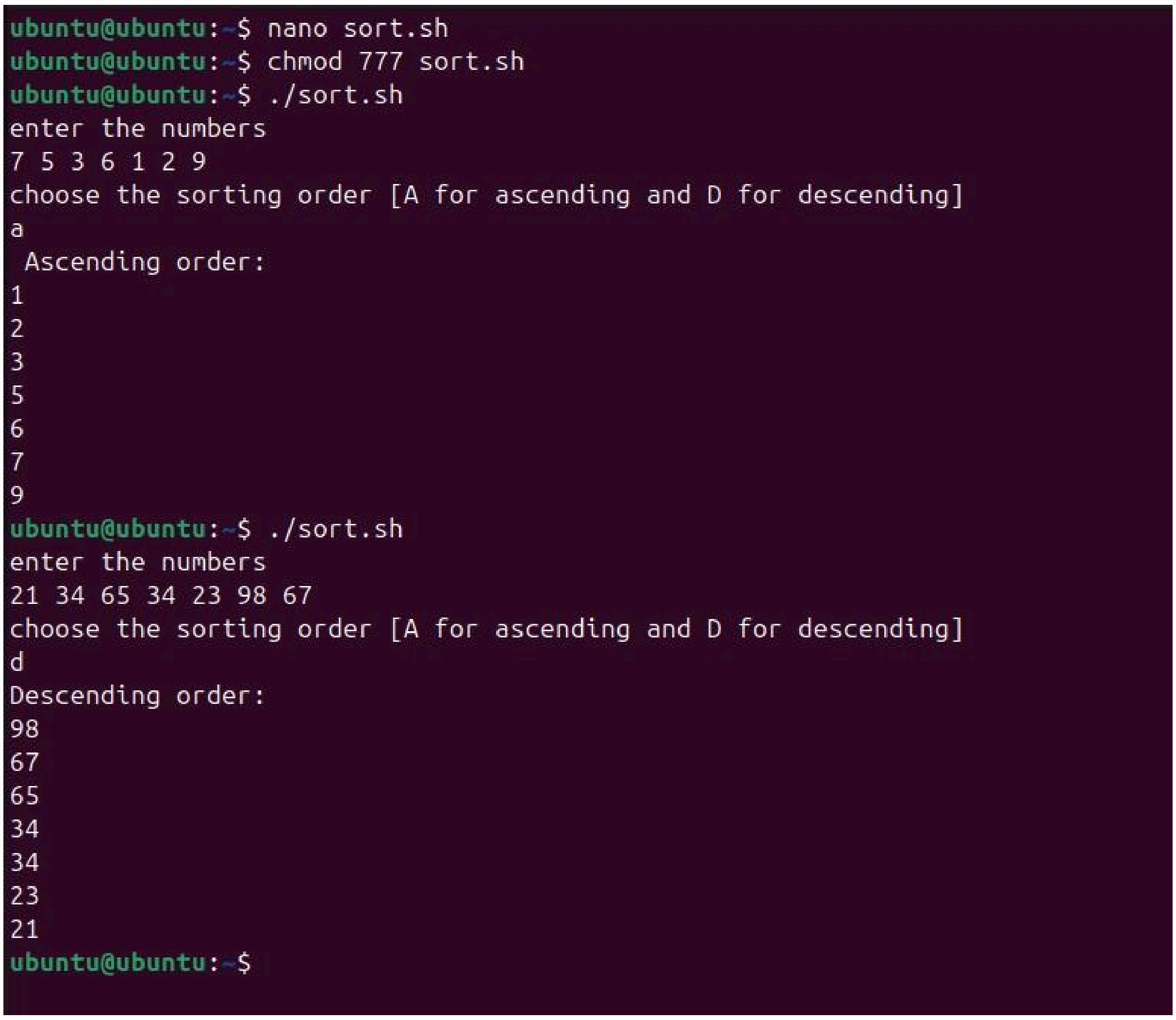


```
GNU nano 7.2
echo "enter the numbers"
read -a num

echo "choose the sorting order [A for ascending and D for descending]"
read order

if [ "$order" = "a" ] || [ "$order" = "A" ];
then
        echo " Ascending order:"
        sorted=$(printf "%s\n" "${num[@]}" | sort -n)
        echo "$sorted"
elif [ "$order" = "d" ] || [ "$order" = "D" ]; then
        echo "Descending order:"
        sorted=$(printf "%s\n" "${num[@]}" | sort -nr)
        echo "$sorted"
else
        echo "invalid option"
fi
```

OUTPUT:



```
ubuntu@ubuntu:~$ nano sort.sh
ubuntu@ubuntu:~$ chmod 777 sort.sh
ubuntu@ubuntu:~$ ./sort.sh
enter the numbers
7 5 3 6 1 2 9
choose the sorting order [A for ascending and D for descending]
a
 Ascending order:
1
2
3
5
6
7
9
ubuntu@ubuntu:~$ ./sort.sh
enter the numbers
21 34 65 34 23 98 67
choose the sorting order [A for ascending and D for descending]
d
Descending order:
98
67
65
34
34
23
21
ubuntu@ubuntu:~$
```

Experiment 7: Shell Programming

i. Write a script that takes a filename as input and checks if it exists. If the file exists, display its content; otherwise, prompt the user to create the file.

SOLUTION:

File Existence Check

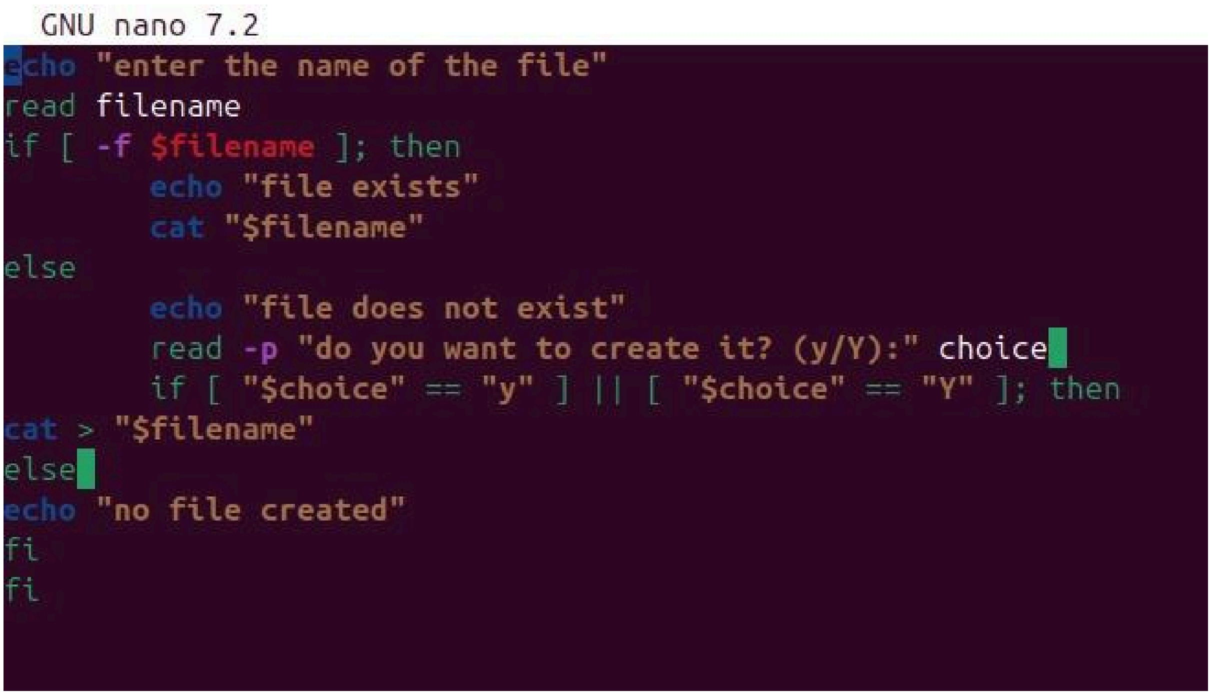
- Take a **filename** as input from the user.
- Use a function or method to **check if the file exists** in the system.

If the File Exists

- Open the file in **read mode**.
- Read and **display its contents** to the user.

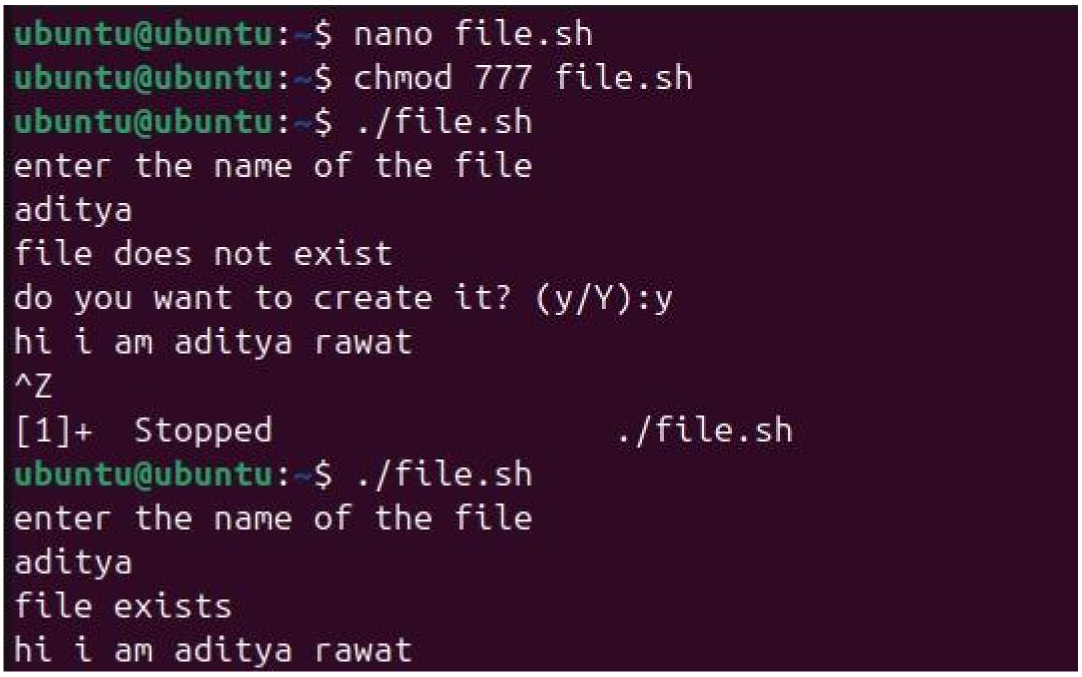
If the File Does Not Exist

- Inform the user that the file doesn't exist.
- Prompt the user to **create the file**.
- If the user agrees:
    - Ask for the **content** to write into the file.
    - Open the file in **write mode** and save the content.
- Display a success message after reading or creating the file.



```
GNU nano 7.2
echo "enter the name of the file"
read filename
if [ -f $filename ]; then
        echo "file exists"
        cat "$filename"
else
        echo "file does not exist"
        read -p "do you want to create it? (y/Y):" choice
        if [ "$choice" == "y" ] || [ "$choice" == "Y" ]; then
cat > "$filename"
else
echo "no file created"
fi
fi
```

OUTPUT:



```
ubuntu@ubuntu:~$ nano file.sh
ubuntu@ubuntu:~$ chmod 777 file.sh
ubuntu@ubuntu:~$ ./file.sh
enter the name of the file
aditya
file does not exist
do you want to create it? (y/Y):y
hi i am aditya rawat
^Z
[1]+  Stopped                 ./file.sh
ubuntu@ubuntu:~$ ./file.sh
enter the name of the file
aditya
file exists
hi i am aditya rawat
```

ii. Create a script that prints the numbers from 1 to 10 using a loop.

SOLUTION:

Looping Logic

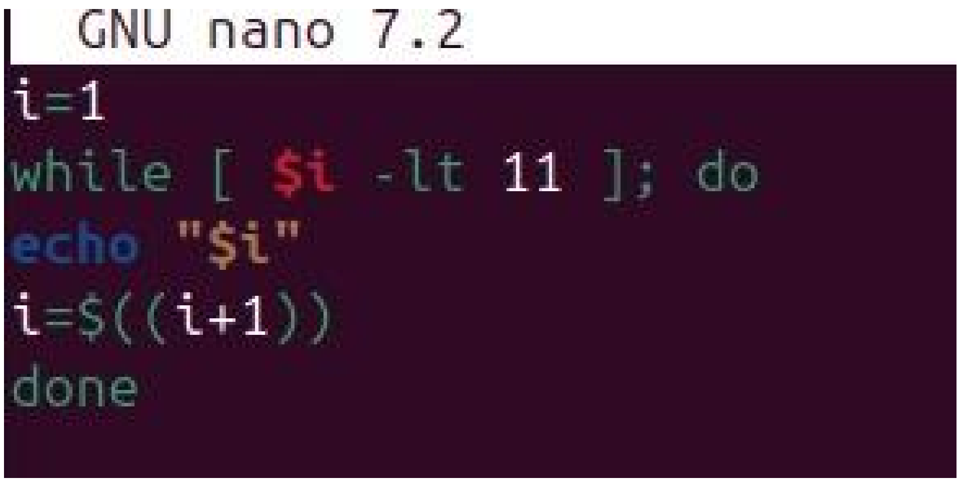
- Use a **loop structure** (e.g., `for` or `while` loop).
- Start the loop at **1** and end at **10** (inclusive).

Iteration

- On each iteration, **print the current number**.

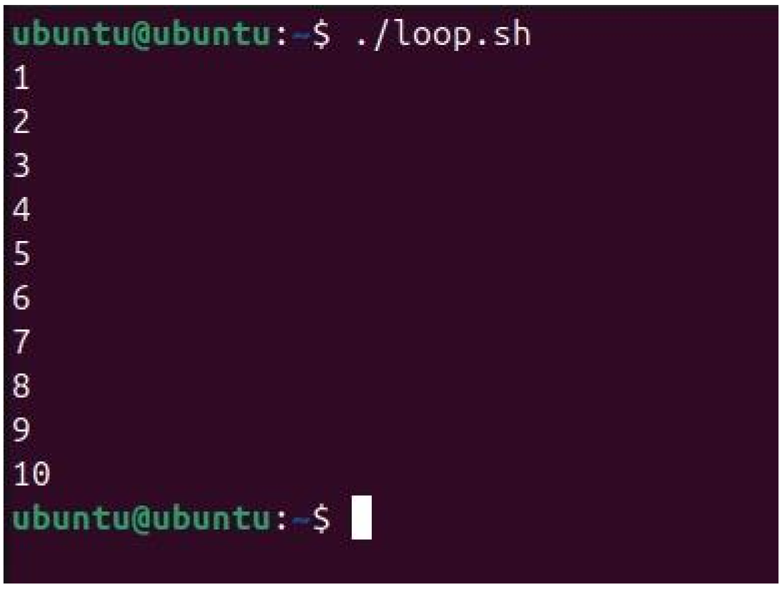
Final Step

- The loop stops automatically after reaching **10**.



```
  GNU nano 7.2
i=1
while [ $i -lt 11 ]; do
echo "$i"
i=$((i+1))
done
```

OUTPUT:



```
ubuntu@ubuntu:~$ ./loop.sh
1
2
3
4
5
6
7
8
9
10
ubuntu@ubuntu:~$ 
```

iii. Write a script that takes a filename as a command line argument and counts the number of lines, words, and characters in that file.

SOLUTION:

- The script should accept **one argument**: the filename.
- This filename should refer to a **text file** present in the system.

- Before proceeding, the script should check whether the given file **exists** and is **readable**.
- If the file doesn't exist or is not readable, the script should **print an error message** and **exit**.

- Linux has a built-in command called `wc` (**word count**) that can:
  - Count **lines** (`-l`)

    - Count **words** (`-w`)

    - Count **characters** (`-m`)

- These options can be used **individually or together** to get the desired output.

- `wc` outputs all three counts in a formatted line if no specific option is given.

- The script can **capture** this output and **extract** each count separately if needed.

- After calculating the counts, the script should print:

    - Number of lines

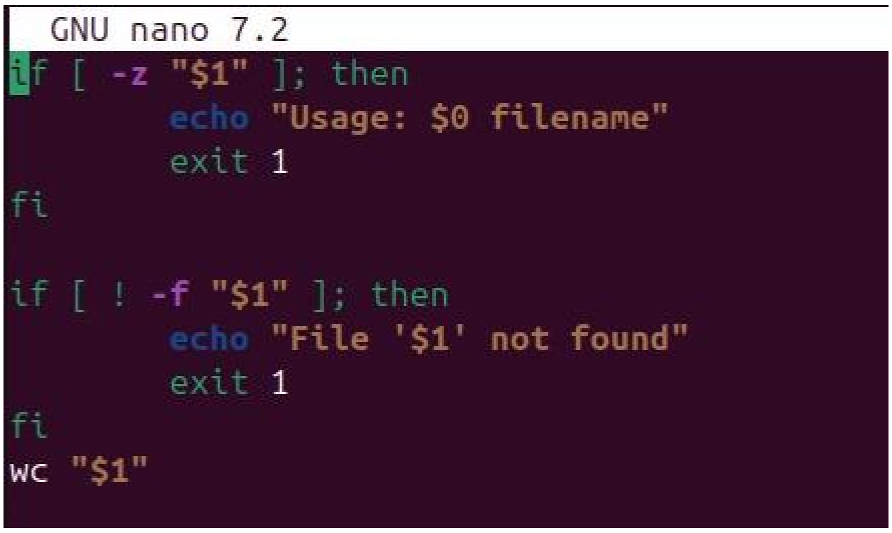
    - Number of words

    - Number of characters

- Each on a separate line, labeled clearly.

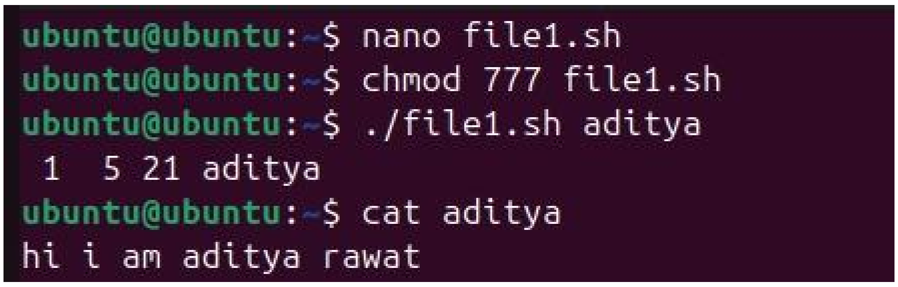
- File is empty → all counts should be zero.

- File contains special characters or multiple spaces → word count should still be accurate (handled by `wc`).


```
  GNU nano 7.2
if [ -z "$1" ]; then
        echo "Usage: $0 filename"
        exit 1
fi

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

OUTPUT:


```
ubuntu@ubuntu:~$ nano file1.sh
ubuntu@ubuntu:~$ chmod 777 file1.sh
ubuntu@ubuntu:~$ ./file1.sh aditya
 1  5 21 aditya
ubuntu@ubuntu:~$ cat aditya
hi i am aditya rawat
```

iv. Create a script that defines a function to calculate the factorial of a given number and call that function with different inputs.

SOLUTION:

Understand Factorial

- Factorial of a number *n* (n!) is the product of all positive integers from 1 to *n*.

- Special case: 0! = 1.

Script Needs a Function

- Define a reusable function inside the script to calculate factorial for any given input.
- Functions in shell scripting help organize code and avoid repetition.

Function Input

- The function should accept one parameter: the number whose factorial is to be calculated.

Calculation Logic

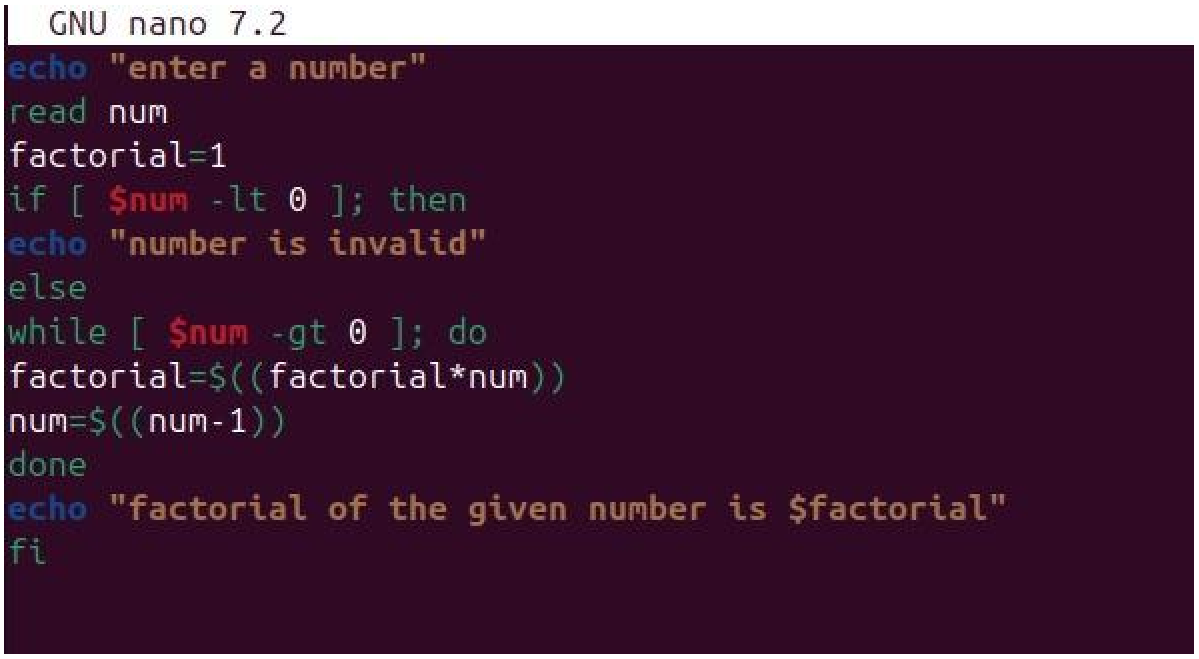
- Use a loop to multiply numbers starting from 1 up to the input number.
- Accumulate the product to get the factorial.

Edge Cases & Validation

- Ensure the input is a non-negative integer (factorial isn’t defined for negatives or decimals).
- Handle input = 0 separately if needed (since factorial of 0 is 1).

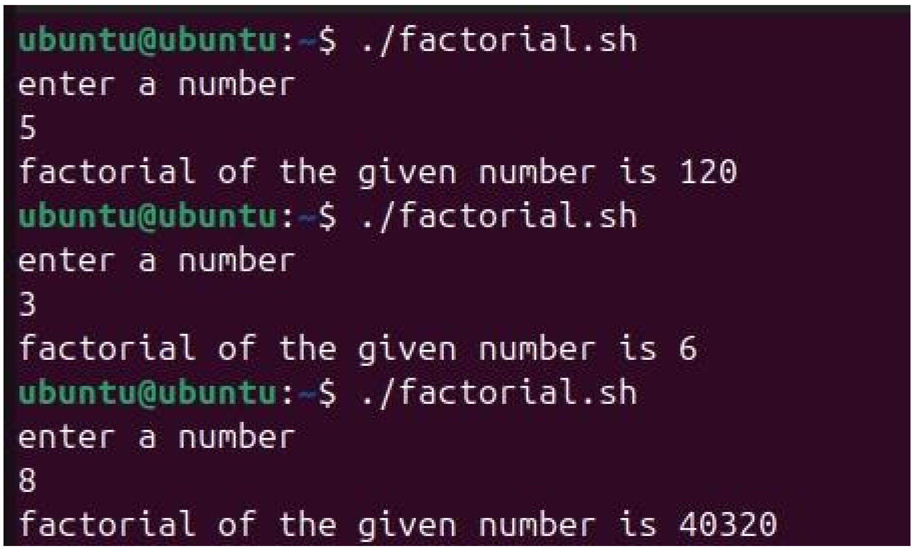
Calling the Function

- After defining the function, call it multiple times in the script with different input values to test and demonstrate functionality.



```
  GNU nano 7.2
echo "enter a number"
read num
factorial=1
if [ $num -lt 0 ]; then
echo "number is invalid"
else
while [ $num -gt 0 ]; do
factorial=$((factorial*num))
num=$((num-1))
done
echo "factorial of the given number is $factorial"
fi
```

OUTPUT:



```
ubuntu@ubuntu:~$ ./factorial.sh
enter a number
5
factorial of the given number is 120
ubuntu@ubuntu:~$ ./factorial.sh
enter a number
3
factorial of the given number is 6
ubuntu@ubuntu:~$ ./factorial.sh
enter a number
8
factorial of the given number is 40320
```