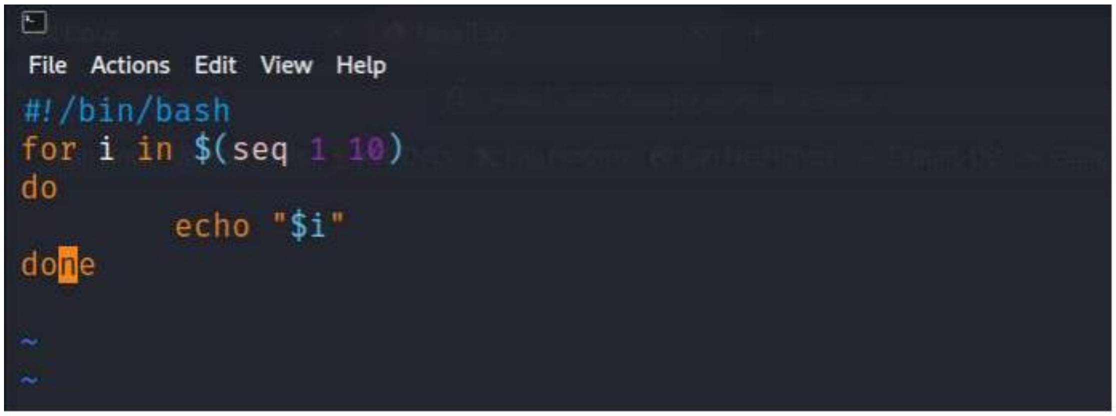
Bash Script

Bsc.Hons Ethical Hacking

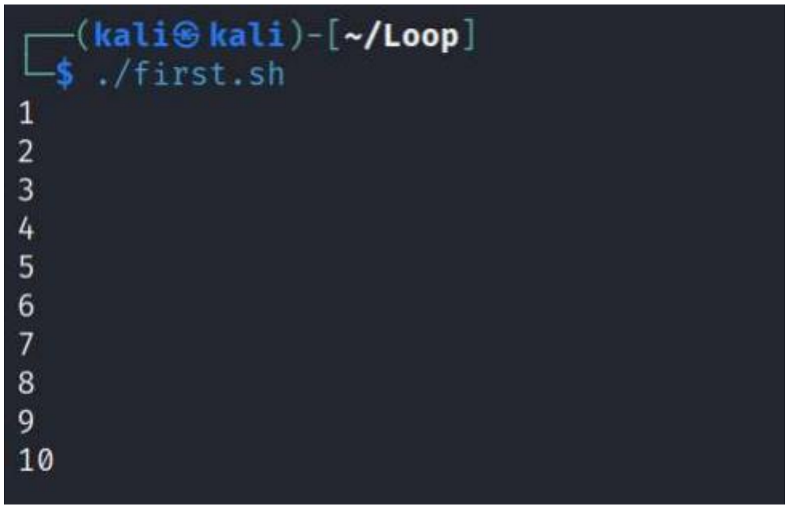
BandanaThapa Magar

Looping:

1. Write a Bash script that uses a for loop to print the numbers from 1 to 10.

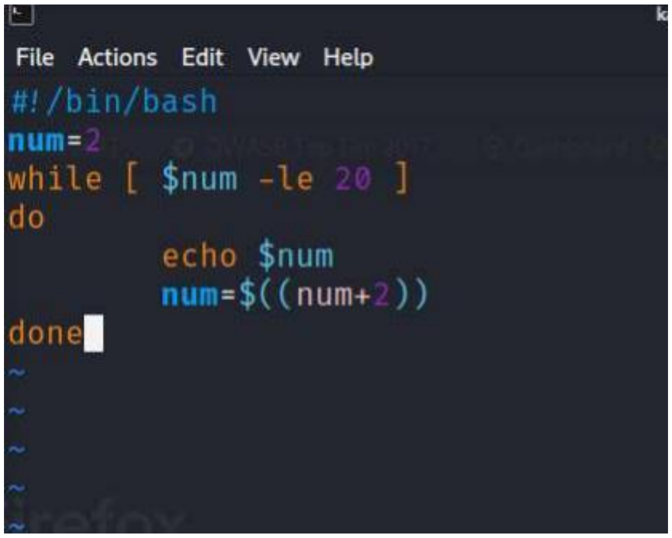


```
#!/bin/bash
for i in $(seq 1 10)
do
        echo "$i"
done

~
~
```

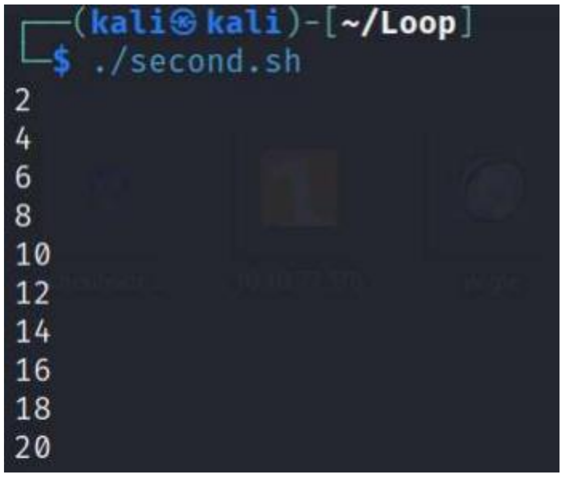


```
┌──(kali㉿kali)-[~/Loop]
└─$ ./first.sh
1
2
3
4
5
6
7
8
9
10
```

2. Create a Bash script that uses a while loop to print the even numbers from 2 to 20.



```
#!/bin/bash
num=2
while [ $num -le 20 ]
do
        echo $num
        num=$((num+2))
done
~
~
~
~
~
```

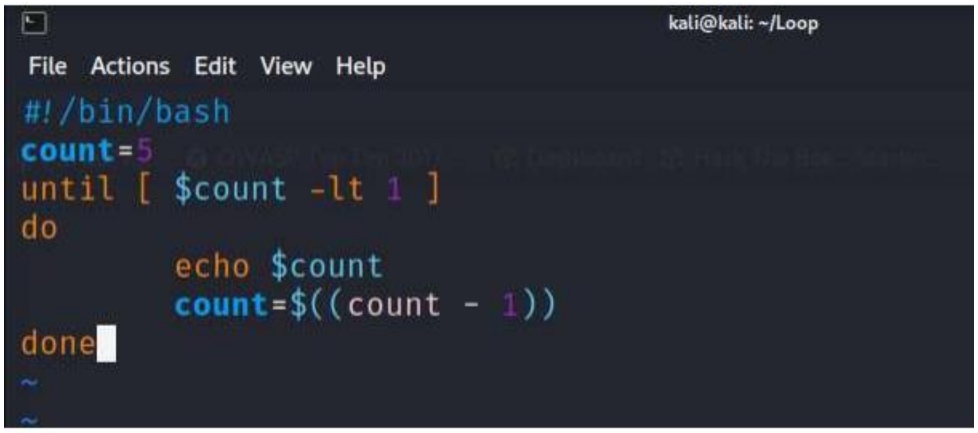


```
┌──(kali㉿kali)-[~/Loop]
└─$ ./second.sh
2
4
6
8
10
12
14
16
18
20
```

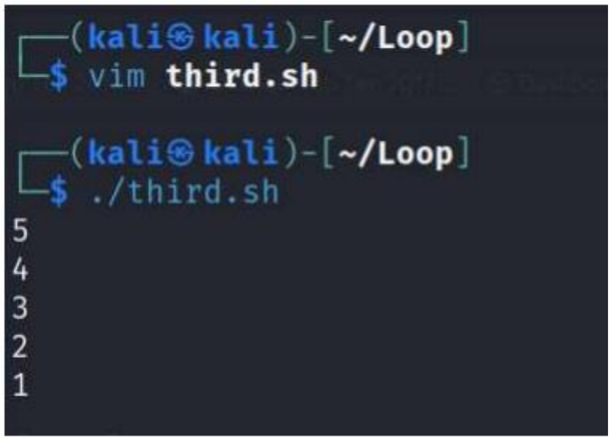
3. Explain the difference between a for loop and a while loop in Bash scripting.

- **For Loop:** A for loop is used when you know in advance how many times you want the loop to iterate. It typically iterates over a sequence, such as a range of numbers or elements in an array, for a fixed number of times.
- **While Loop:** A while loop is used when you want to repeat a set of commands as long as a certain condition is true. It keeps looping until the specified condition becomes false.

4. Write a Bash script that uses a until loop to count down from 5 to 1.



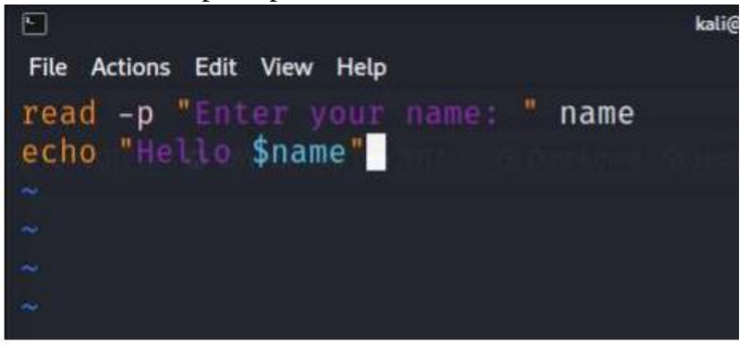
```
#!/bin/bash
count=5
until [ $count -lt 1 ]
do
        echo $count
        count=$((count - 1))
done
~
~
```



```
┌──(kali㉿kali)-[~/Loop]
└─$ vim third.sh

┌──(kali㉿kali)-[~/Loop]
└─$ ./third.sh
5
4
3
2
1
```

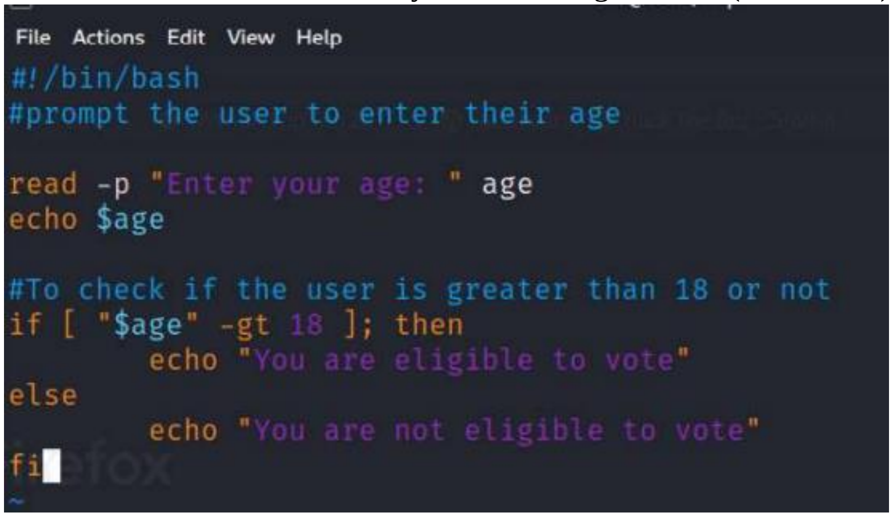
Asking Input:

1. How do we prompt a user for their name in a Bash script and store it in a variable?

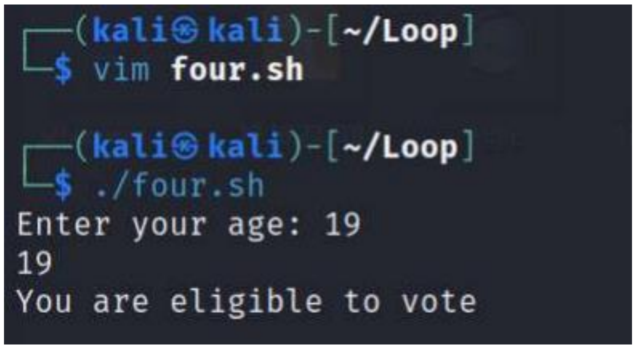


```
read -p "Enter your name: " name
echo "Hello $name"
~
~
~
~
```

2. Write a Bash script that asks the user for their age and then displays a message based on whether they are old enough to vote (18 or older).



```
#!/bin/bash
#prompt the user to enter their age

read -p "Enter your age: " age
echo $age

#To check if the user is greater than 18 or not
if [ "$age" -gt 18 ]; then
        echo "You are eligible to vote"
else
        echo "You are not eligible to vote"
fi
~
```



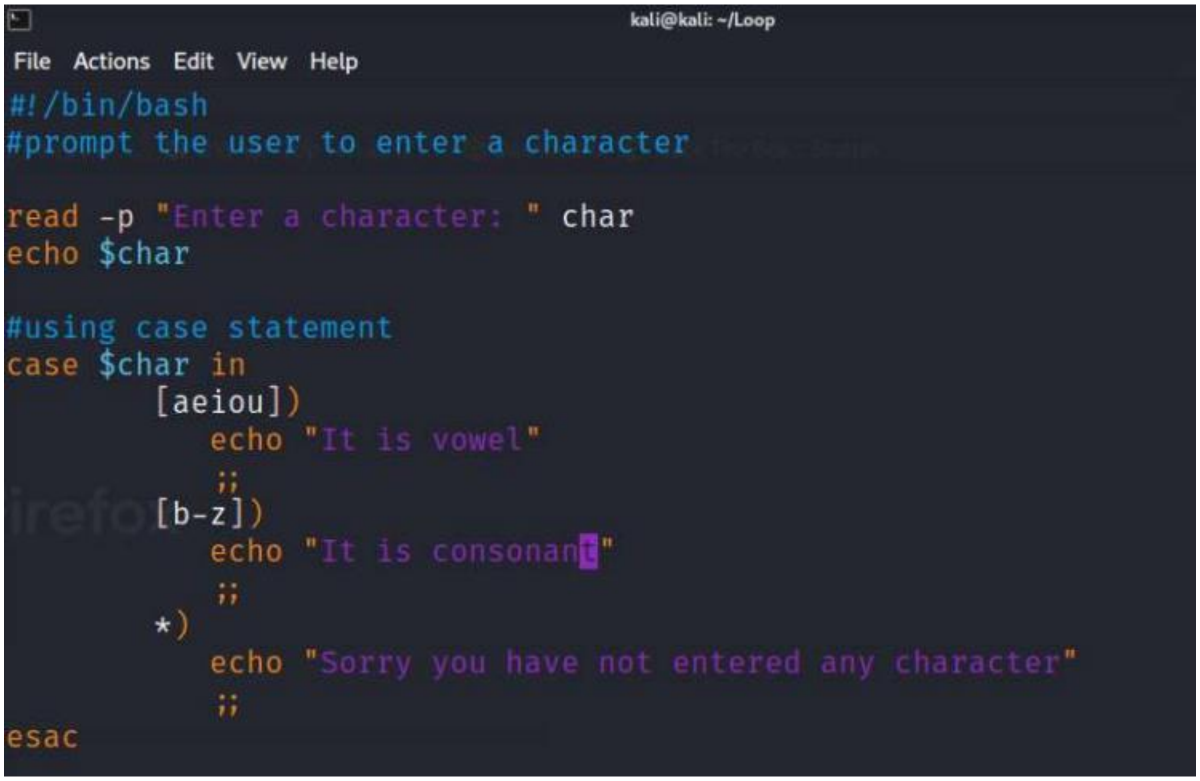
```
┌──(kali㉿kali)-[~/Loop]
└─$ vim four.sh

┌──(kali㉿kali)-[~/Loop]
└─$ ./four.sh
Enter your age: 19
19
You are eligible to vote
```

Case Conditional Statement:

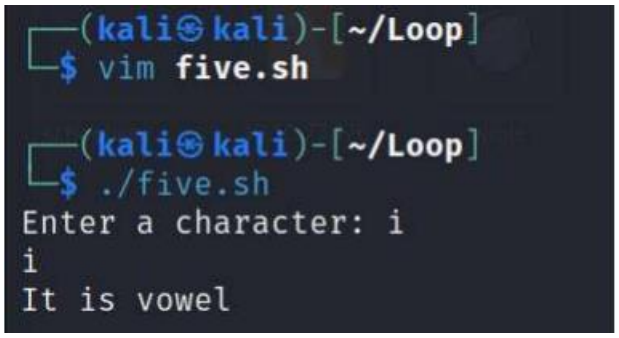
1. Explain the purpose of the case statement in Bash scripting.

- ➢ The case statement is used for making decisions in scripts. It works like upgraded version of if-else statement. Its main function is to streamline the task of checking a single variable or value against several potential values or patterns.

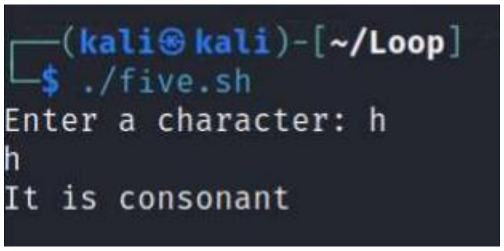
2. Create a Bash script that uses a case statement to determine if a given input is a vowel or a consonant.



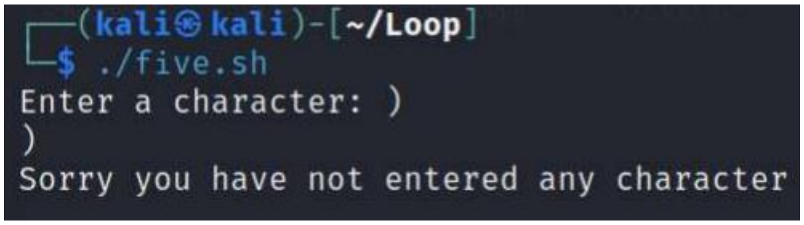
```
#!/bin/bash
#prompt the user to enter a character

read -p "Enter a character: " char
echo $char

#using case statement
case $char in
        [aeiou])
           echo "It is vowel"
           ;;
        [b-z])
           echo "It is consonant"
           ;;
        *)
           echo "Sorry you have not entered any character"
           ;;
esac
```



```
┌──(kali㉿kali)-[~/Loop]
└─$ vim five.sh

┌──(kali㉿kali)-[~/Loop]
└─$ ./five.sh
Enter a character: i
i
It is vowel
```



```
┌──(kali㉿kali)-[~/Loop]
└─$ ./five.sh
Enter a character: h
h
It is consonant
```



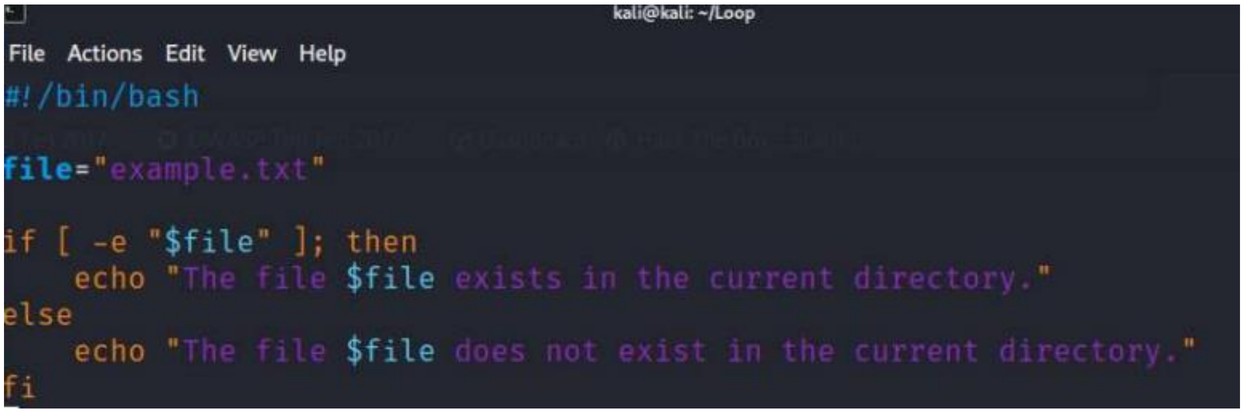
```
┌──(kali㉿kali)-[~/Loop]
└─$ ./five.sh
Enter a character: )
)
Sorry you have not entered any character
```

3. What is the significance of the ;; in a case statement?

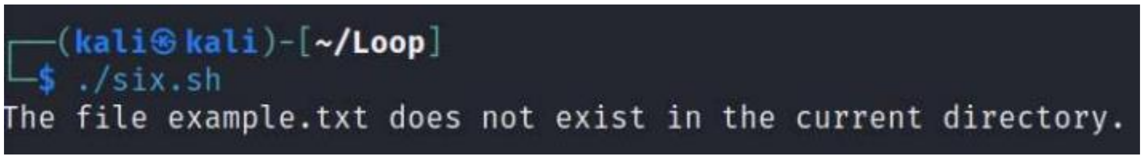
➢ In case statement;; plays its role as a separator between different code blocks associated with different patterns. Basically, it tells bash to stop the processing the current code and exit the case statement once a matching pattern is found.

Conditional Statements:

1. Write a Bash script that uses an if statement to check if a file named "example.txt" exists in the current directory.

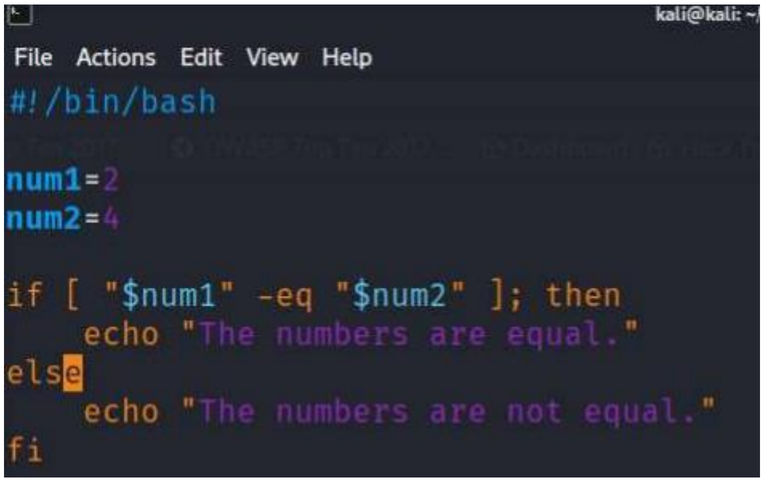


```
#!/bin/bash

file="example.txt"

if [ -e "$file" ]; then
    echo "The file $file exists in the current directory."
else
    echo "The file $file does not exist in the current directory."
fi
```

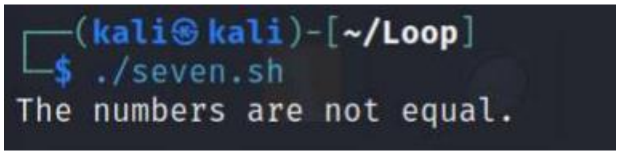


```
┌──(kali㉿kali)-[~/Loop]
└─$ ./six.sh
The file example.txt does not exist in the current directory.
```

2. How do you use the -eq operator to compare two numbers in a Bash if statement?

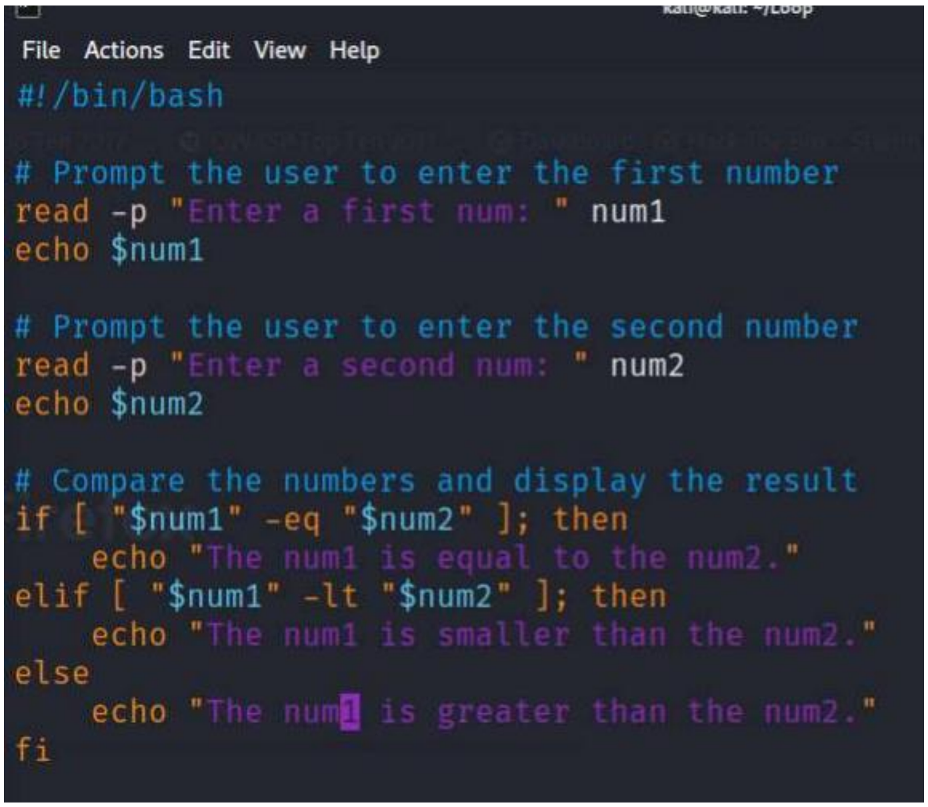


```
#!/bin/bash

num1=2
num2=4

if [ "$num1" -eq "$num2" ]; then
    echo "The numbers are equal."
else
    echo "The numbers are not equal."
fi
```

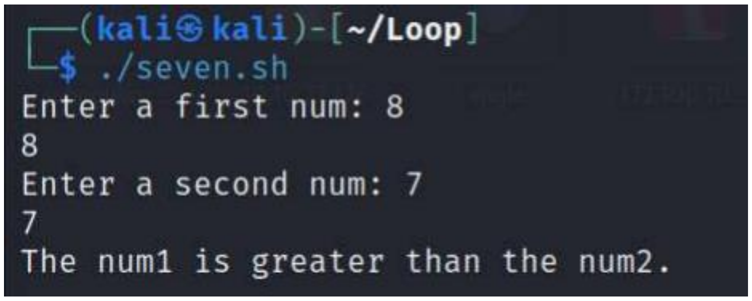


```
┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
The numbers are not equal.
```

3. Write a Bash script that compares two numbers entered by the user and displays whether the first number is greater, smaller, or equal to the second number.



```bash
#!/bin/bash

# Prompt the user to enter the first number
read -p "Enter a first num: " num1
echo $num1

# Prompt the user to enter the second number
read -p "Enter a second num: " num2
echo $num2

# Compare the numbers and display the result
if [ "$num1" -eq "$num2" ]; then
    echo "The num1 is equal to the num2."
elif [ "$num1" -lt "$num2" ]; then
    echo "The num1 is smaller than the num2."
else
    echo "The num1 is greater than the num2."
fi
```



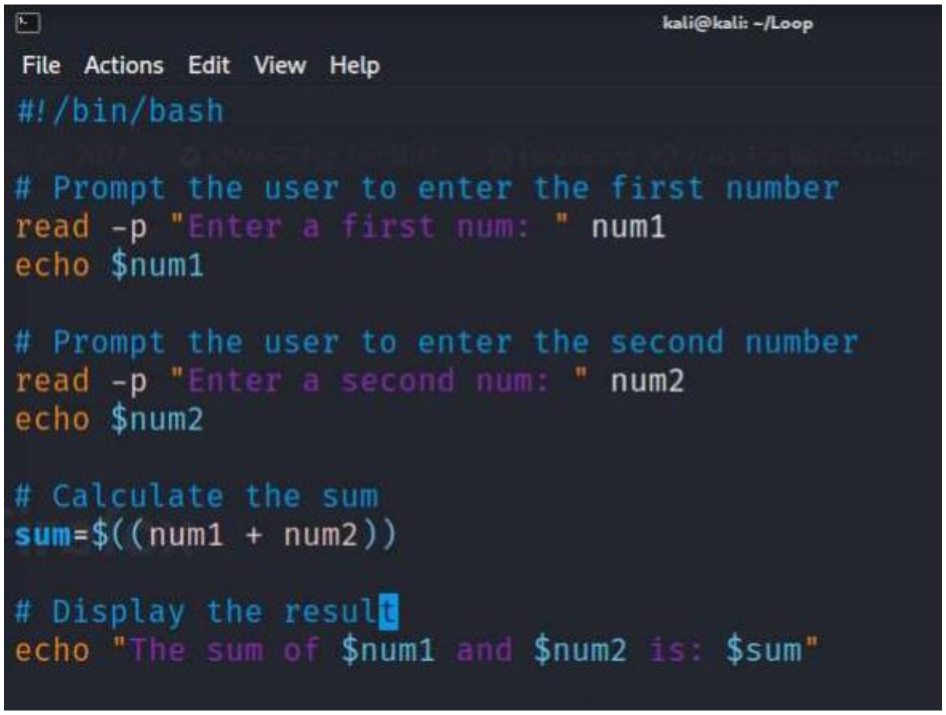
```
┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
Enter a first num: 8
8
Enter a second num: 7
7
The num1 is greater than the num2.
```

Arithmetic:

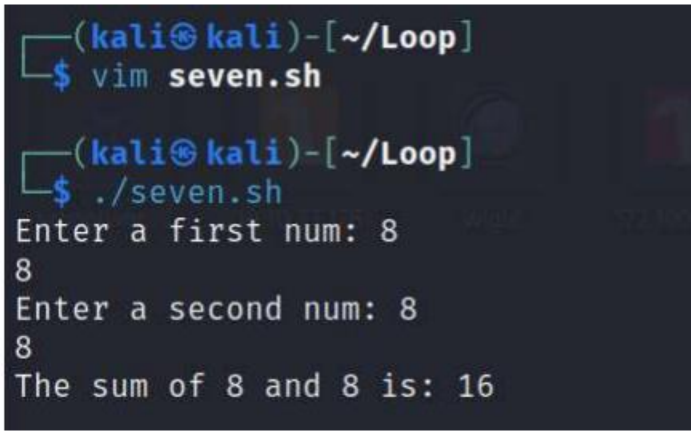
1. Explain the purpose of the expr command in Bash scripting.

➢ The expr command in bash scripting serves as a tool for performing calculations, comparisons, and string operations. Its main function is to facilitate these tasks within the script making it a versatile and flexible utility for various operations.

2. Create a Bash script that calculates the sum of two numbers entered by the user.

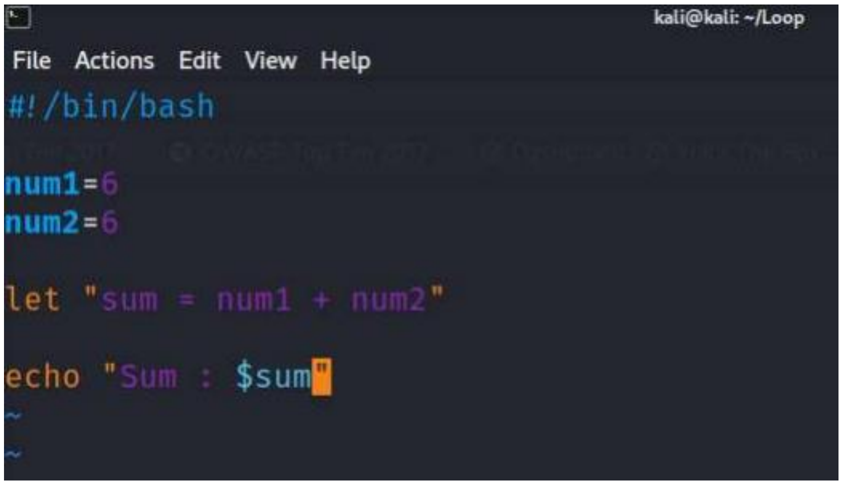


```
#!/bin/bash

# Prompt the user to enter the first number
read -p "Enter a first num: " num1
echo $num1

# Prompt the user to enter the second number
read -p "Enter a second num: " num2
echo $num2

# Calculate the sum
sum=$((num1 + num2))

# Display the result
echo "The sum of $num1 and $num2 is: $sum"
```

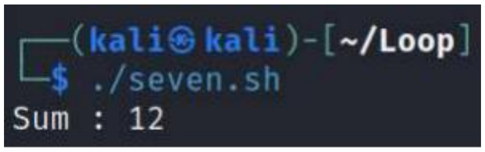


```
┌──(kali㉿kali)-[~/Loop]
└─$ vim seven.sh

┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
Enter a first num: 8
8
Enter a second num: 8
8
The sum of 8 and 8 is: 16
```

3. How do you use the let command to perform arithmetic operations in Bash?



```
#!/bin/bash

num1=6
num2=6

let "sum = num1 + num2"

echo "Sum : $sum"
~
~
```

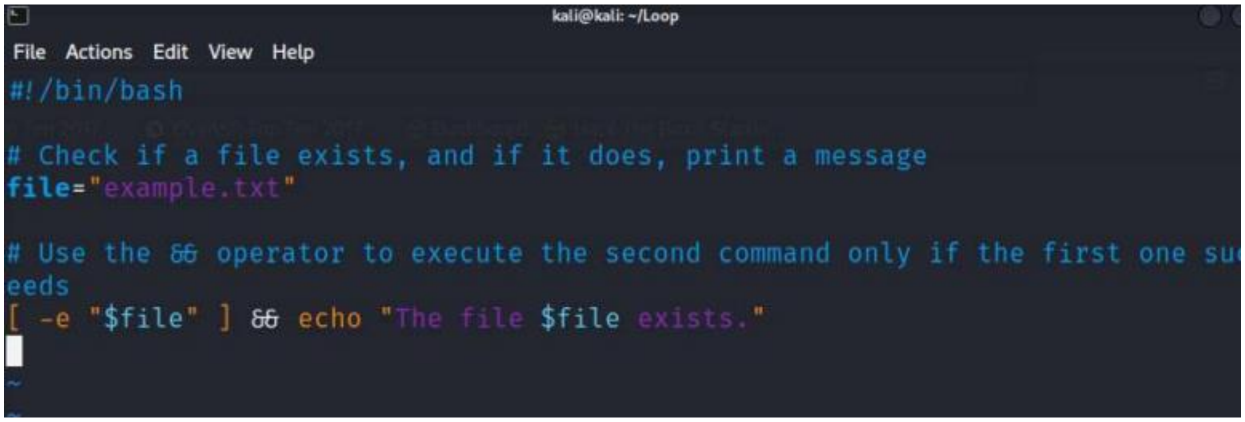


```
┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
Sum : 12
```

Operators:

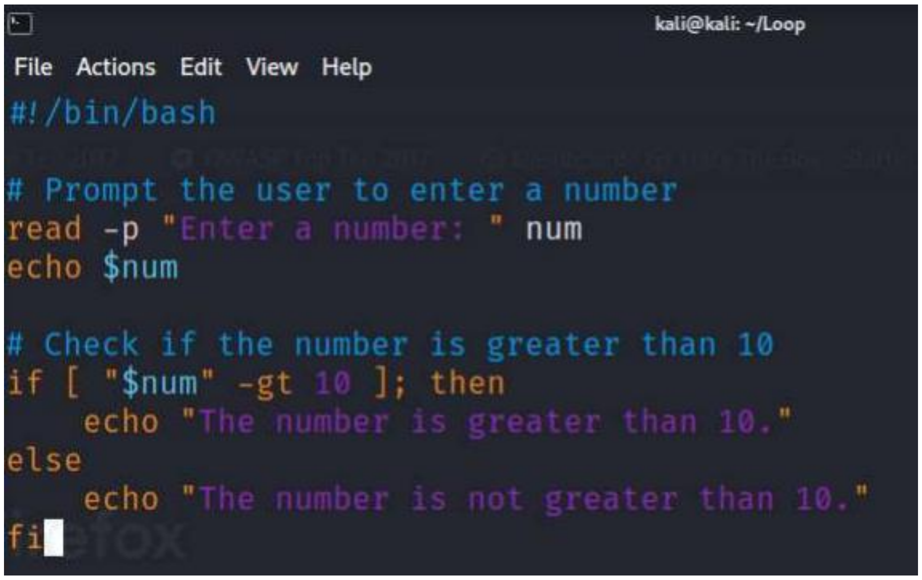
1. What is the purpose of the && operator in Bash scripting? Provide an example.

➢ && operator in bash scripting is used for conditional execution of commands. Here is the example below:

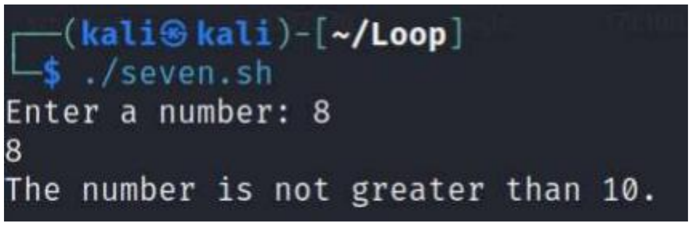


```
#!/bin/bash

# Check if a file exists, and if it does, print a message
file="example.txt"

# Use the && operator to execute the second command only if the first one su
eeds
[ -e "$file" ] && echo "The file $file exists."
```

2. Write a Bash script that uses the -gt operator to check if a number is greater than 10.



```bash
#!/bin/bash

# Prompt the user to enter a number
read -p "Enter a number: " num
echo $num

# Check if the number is greater than 10
if [ "$num" -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is not greater than 10."
fi
```



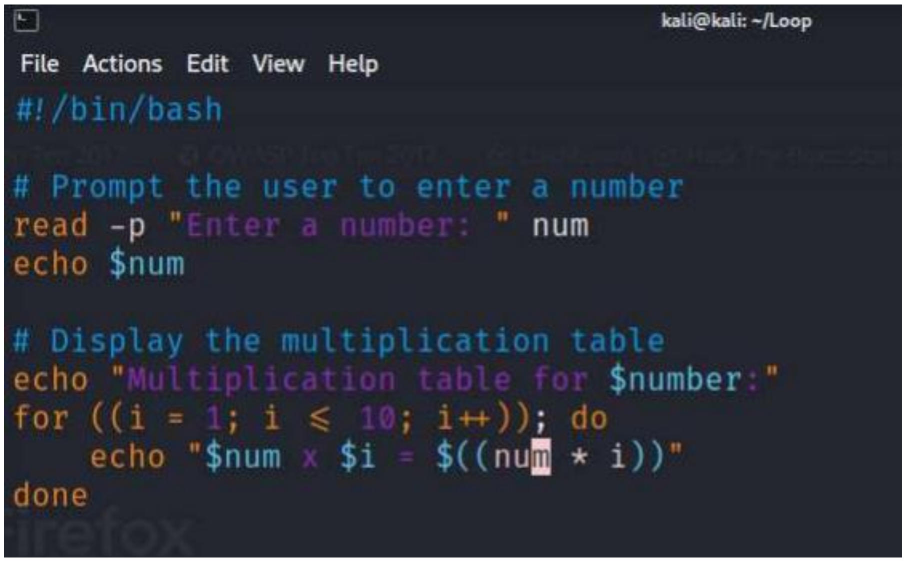
```
┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
Enter a number: 8
8
The number is not greater than 10.
```

3. Explain the difference between = and == in Bash for string comparisons.

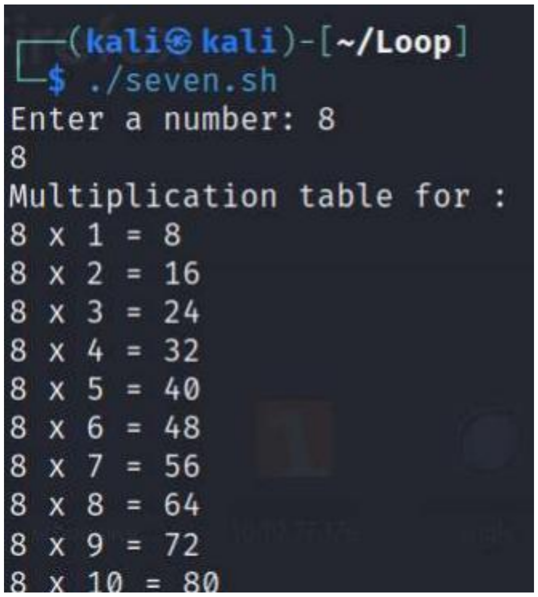
- **=** is used for variable assignment
- == is used for string comparison

Mixed Questions:

1. Create a Bash script that asks the user for a number and then uses a loop to print the multiplication table for that number from 1 to 10.

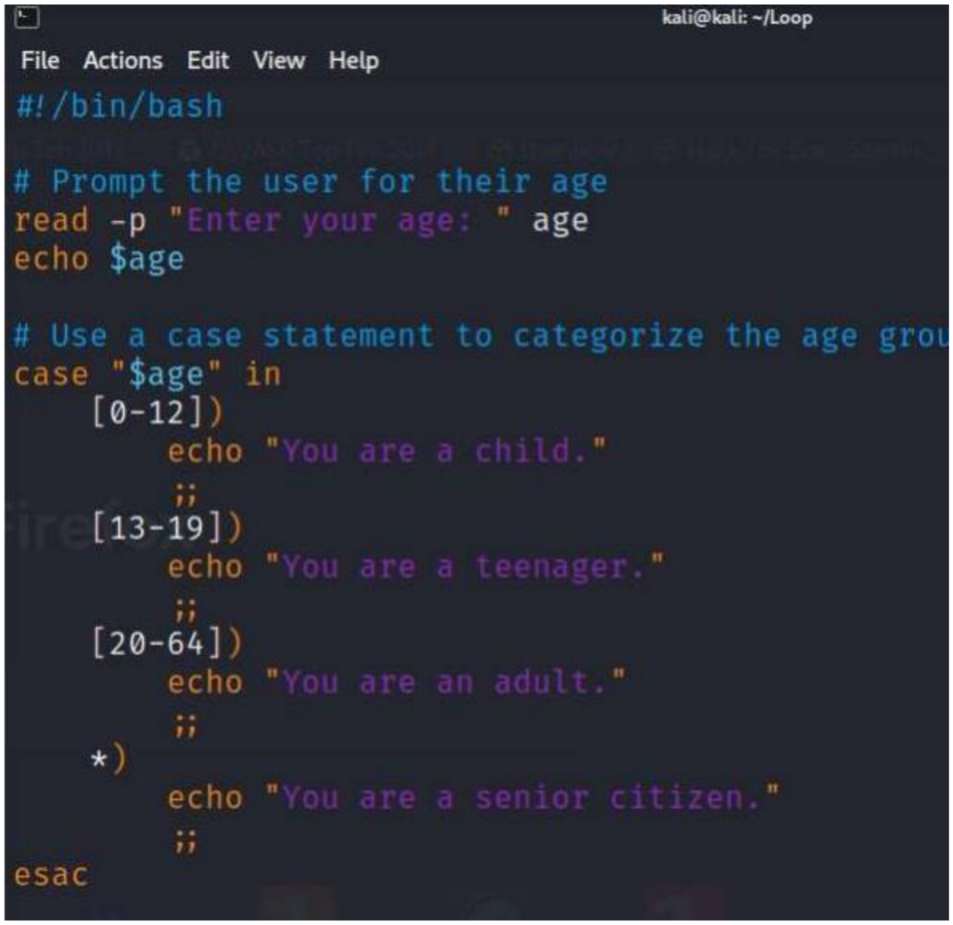


```
#!/bin/bash

# Prompt the user to enter a number
read -p "Enter a number: " num
echo $num

# Display the multiplication table
echo "Multiplication table for $number:"
for ((i = 1; i ≤ 10; i++)); do
    echo "$num x $i = $((num * i))"
done
```

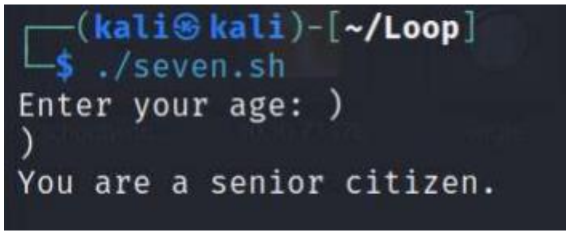


```
┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
Enter a number: 8
8
Multiplication table for :
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

2. Write a Bash script that takes a user's age as input and then uses a case statement to display a message based on whether they are a child, teenager, adult, or senior citizen.



```
#!/bin/bash

# Prompt the user for their age
read -p "Enter your age: " age
echo $age

# Use a case statement to categorize the age grou
case "$age" in
    [0-12])
        echo "You are a child."
        ;;
    [13-19])
        echo "You are a teenager."
        ;;
    [20-64])
        echo "You are an adult."
        ;;
    *)
        echo "You are a senior citizen."
        ;;
esac
```



```
┌──(kali㉿kali)-[~/Loop]
└─$ ./seven.sh
Enter your age: )
)
You are a senior citizen.
```