# Performing Arithmetic Operations in Bash

**Purpose:** Use of double parentheses for arithmetic operations in a Bash shell script

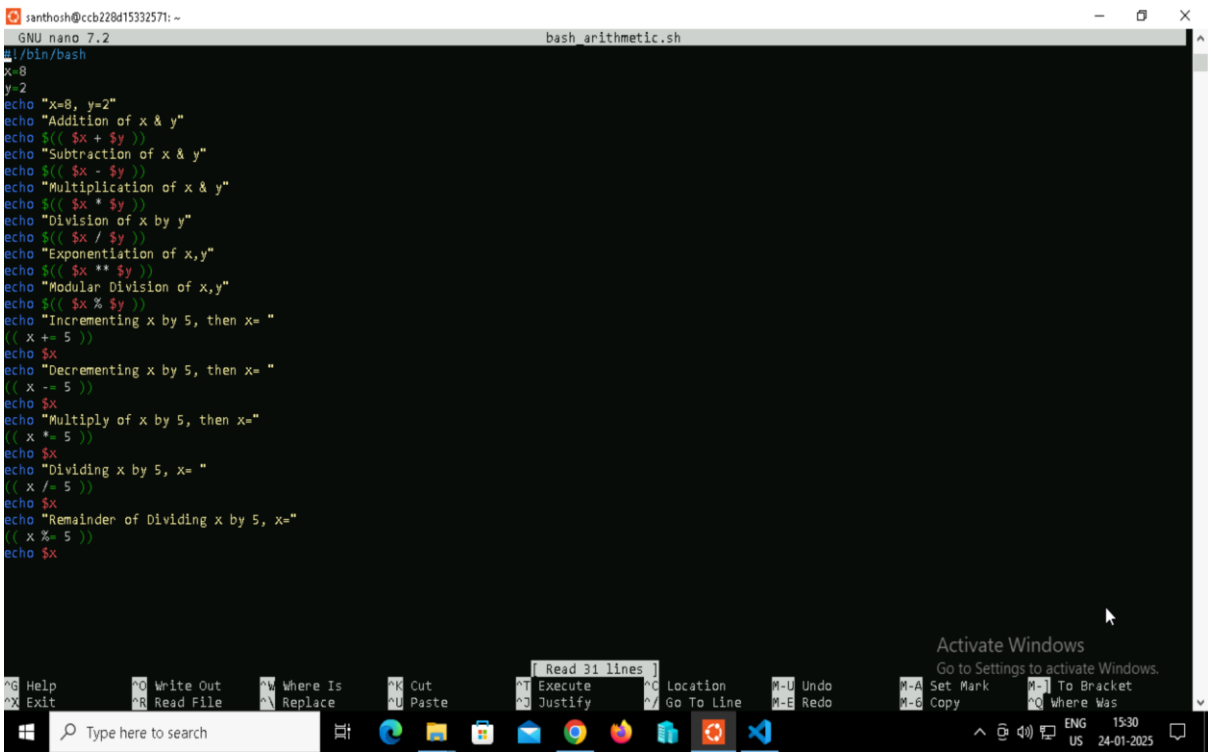
**Step 1:** Touch a file named bash_arithmetic.sh and nano the file to add the bash shell script to the file.

```
santhosh@ccb228d15332571:~$ touch bash_arithmetic.sh
santhosh@ccb228d15332571:~$ nano bash_arithmetic.sh
```

**Step 2:** Using the Double parentheses expression is the easiest mechanism to perform basic arithmetic operations in the Bash shell. We can use this method by using double brackets with or without a leading $.

Script Description:

- The read command with the -p flag displays a prompt (Enter number:) and waits for the user to input a value.
- The value entered by the user is stored in the variable number.
- The if statement evaluates the condition [ $number -gt 125 ].
- -gt is used to check if the number is greater than 125.
- If the condition evaluates to true, the script executes the then block and prints.



**Step 3:** Providing the desired execute permissions to the file.

```
santhosh@ccb228d15332571:~$ chmod +x bash_arithmetic.sh
```

**Step 4:** Running the file script using "./ file name" command.

```
santhosh@ccb228d15332571:~$ ./bash_arithmetic.sh
```

**Step 5:** Output

```
X=8, y=2
Addition of x & y
10
Subtraction of x & y
6
Multiplication of x & y
16
Division of x by y
4
Exponentiation of x,y
64
Modular Division of x,y
0
Incrementing x by 5, then X=
13
Decrementing x by 5, then X=
8
Multiply of x by 5, then X=
40
Dividing x by 5, X=
8
Remainder of Dividing x by 5, X=
3
santhosh@ccb228d15332571:~$ nano bash_arithmetic.sh
```

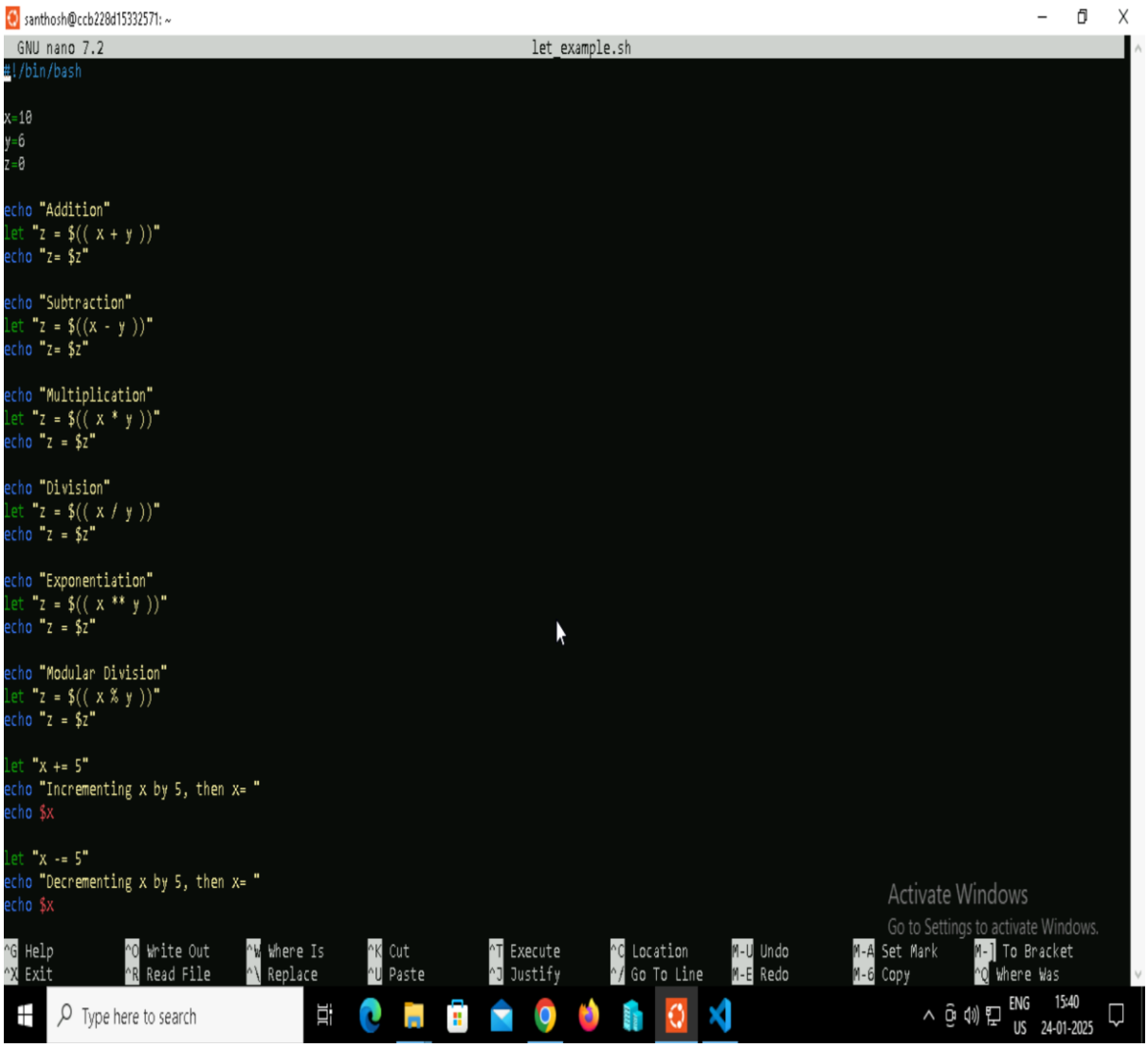
## Let Construction

Let is a built-in command of Bash that allows us to perform arithmetic operations.

**Step 1:** Touch a file named let_example.sh and nano the file to add the bash shell script to the file.

```
santhosh@ccb228d15332571:~$ touch let_example.sh
santhosh@ccb228d15332571:~$ nano let_example.sh
```

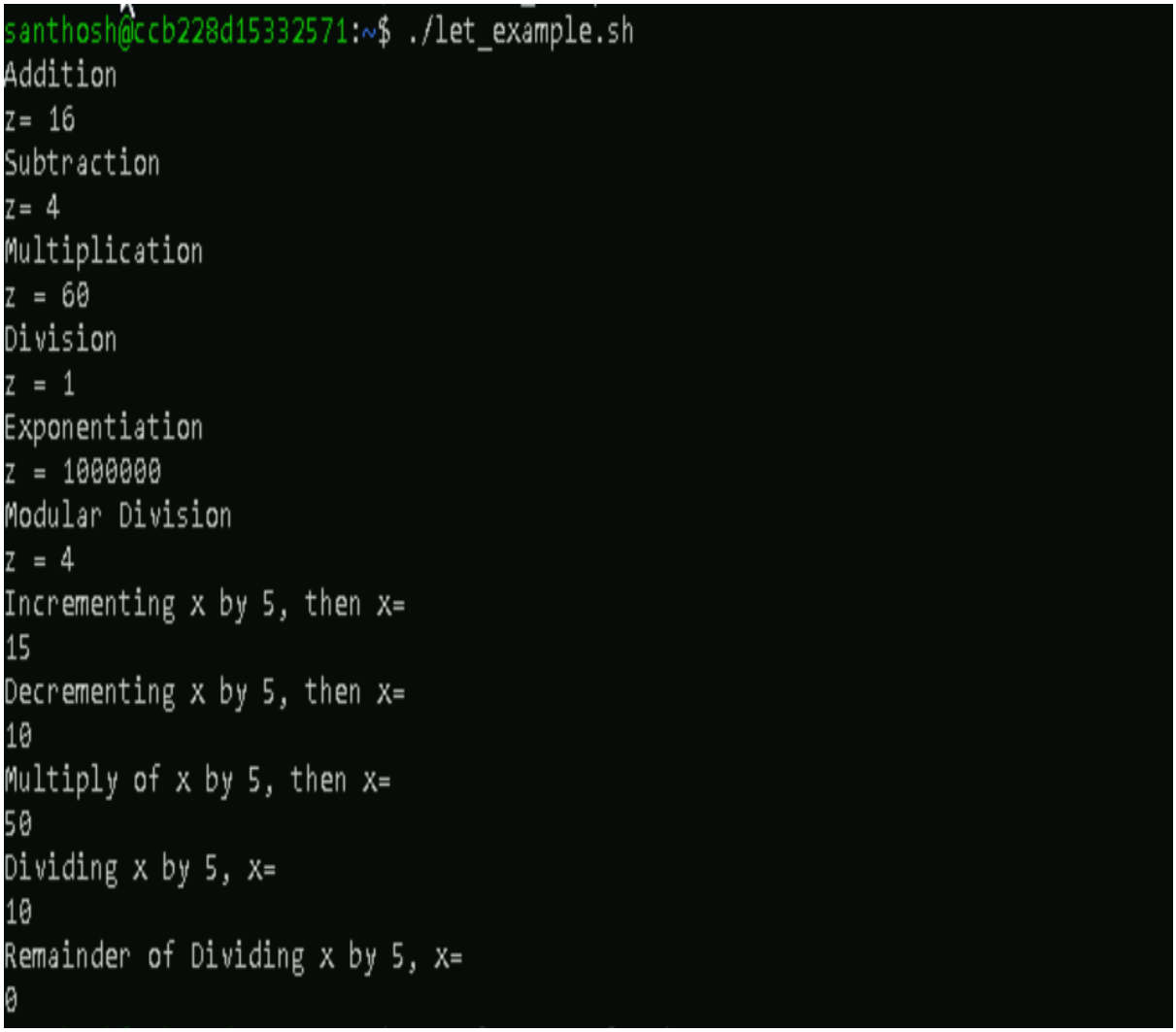
**Step 2:** Adding the Let Construction Script to the file.




**Step 3:** Providing the desired execute permissions to the file.

```
santhosh@ccb228d15332571:~$ chmod +x let_example.sh
```

**Step 4:** Running the file script using "./ file name" command and output.



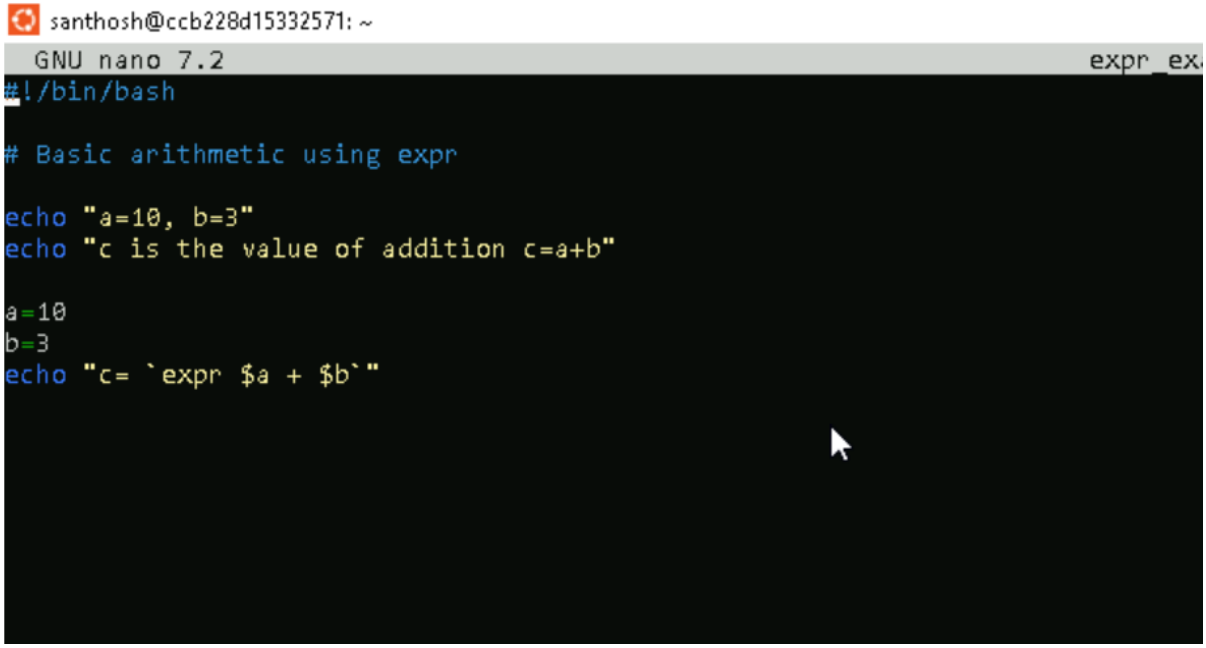
# BACKTICKS

In bash scripting, an arithmetic expansion can also be performed using backticks and expr (known as all-purpose expression 1evaluator).

**Step 1:** Touch a file named expr_example.sh and nano the file to add the bash shell script to the file.

```
santhosh@ccb228d15332571:~$ touch expr_example.sh
santhosh@ccb228d15332571:~$ nano expr_example.sh
```

**Step 2:** Adding the Backtips Script to the file.




**Step 3:** Providing the desired execute permissions to the file.

```
santhosh@ccb228d15332571:~$ chmod +x expr_example.sh
```

**Step 4:** Running the file script using "./ file name" command and output.

```
santhosh@ccb228d15332571:~$ ./expr_example.sh
a=10, b=3
c is the value of addition c=a+b
c= 13
```

# BASH -IF

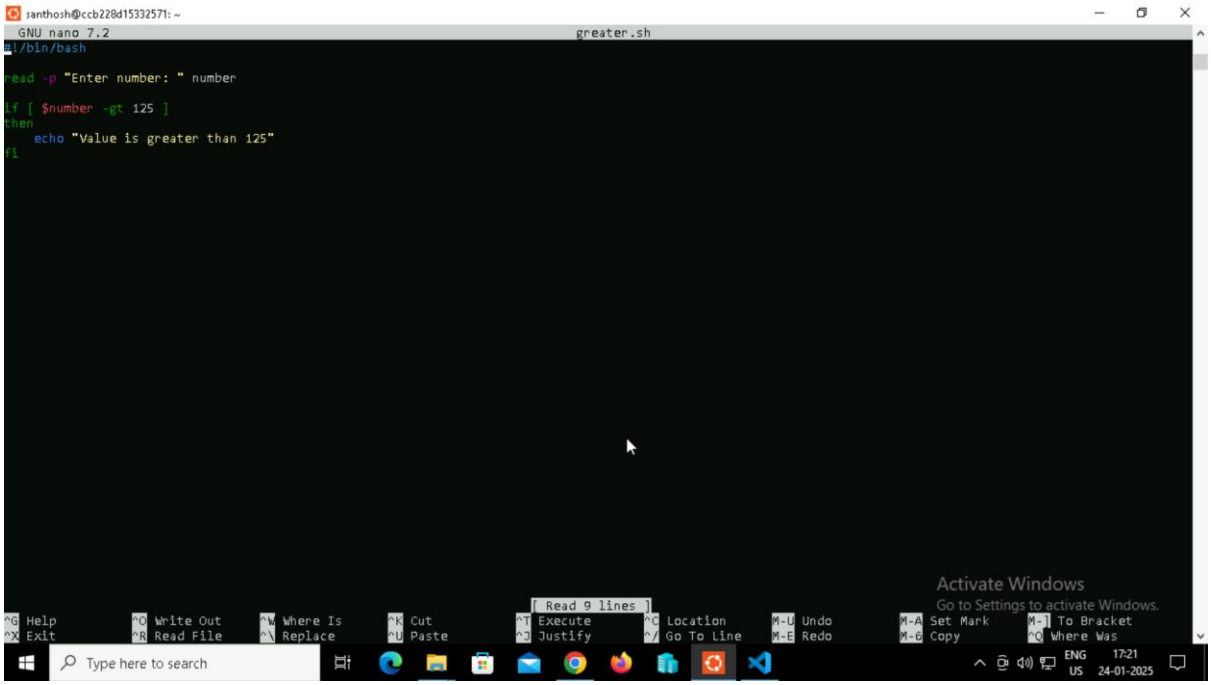
## Example 1

**In this example, take a user-input of any number and check if the value is greater than 125.**

**Step 1:** Touch the greater.sh file and nano the file to save a shell script inside the greter.sh file.

```
santhosh@ccb228d15332571:~$ touch greater.sh
santhosh@ccb228d15332571:~$ nano greater,sh
santhosh@ccb228d15332571:~$ nano greater.sh
```

**Step 2:** Adding the Bash Script for the greater.sh




**Step 3:** Providing permissions for the greater.sh file.

```
santhosh@ccb228d15332571:~$ chmod +x greater.sh
```

**Step 4:** Executing the output file.

```
santhosh@ccb228d15332571:~$ ./greater.sh
Enter number: 189
Value is greater than 125
```

# Example 2

## In this example, we demonstrate the usage of if statement with a simple scenario of comparing two strings:

**Step 1:** Creating a file named compare using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch compare.sh
santhosh@ccb228d15332571:~$ nano comapre.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.


**Step 3:** Providing the necessary permissions to the compare.sh file

```
santhosh@ccb228d15332571:~$ chmod +x comapre.sh
```

**Step 4:** Running the compare.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./comapre.sh
true condition
```

# Example 3

## In this example, we demonstrate how to compare numbers by using the if statement:

**Step 1:** Creating a file named num_if.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch num_if.sh
santhosh@ccb228d15332571:~$ nano num_if.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.


**Step 3:** Providing the necessary permissions to the compare.sh file

```
santhosh@ccb228d15332571:~$ chmod +x num_if.sh
```

**Step 4:** Running the compare.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./num_if.sh
10 is greater than 3.
3 is less than 10.
10 is equal to 10.
```

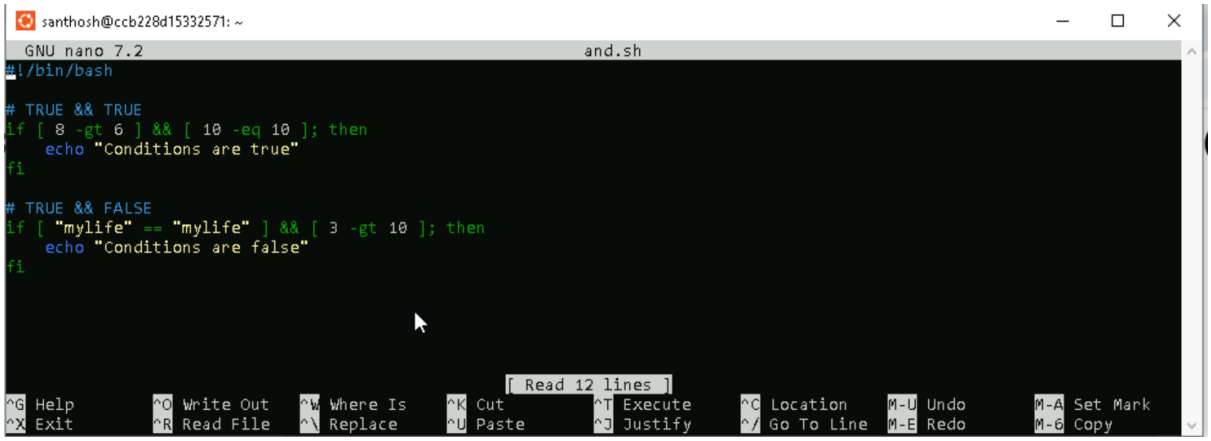
# Example 4

## In this example, we will define how to use AND operator to include multiple conditions in the if expression:

**Step 1:** Creating a file named and.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch and.sh
santhosh@ccb228d15332571:~$ nano and.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the and.sh file

```
santhosh@ccb228d15332571:~$ chmod +x and.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./and.sh
Conditions are true
```

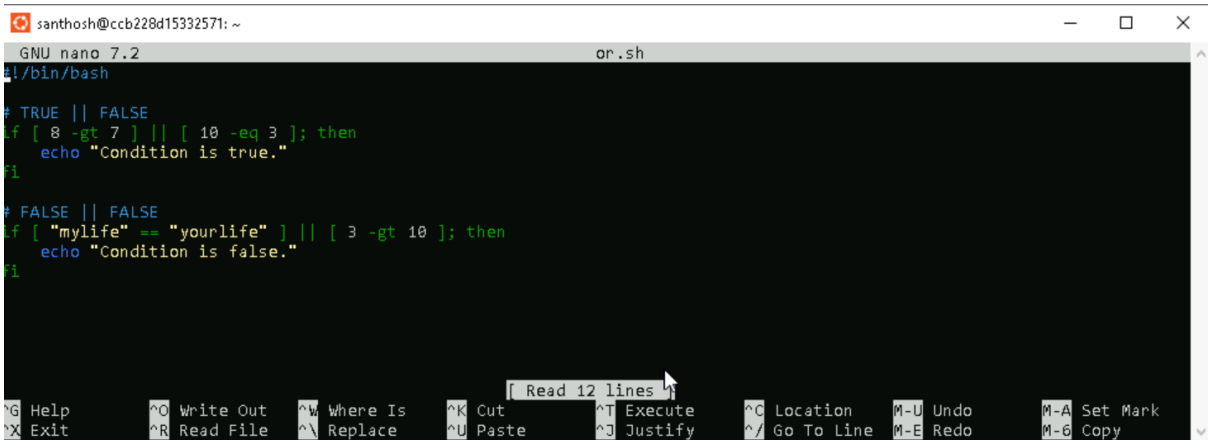
# Example 5

## In this example, we will define how to use OR operator to include multiple conditions in the if expression:

**Step 1:** Creating a file named or.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch or.sh
santhosh@ccb228d15332571:~$ nano or.h
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the or.sh file

```
santhosh@ccb228d15332571:~$ chmod +x or.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./or.sh
Condition is true.
```

# Example 6

## In this example, we will define how to use AND and OR to include multiple conditions in the if expression:

**Purpose:** The script demonstrates the use of logical operators (&&, ||) to evaluate multiple conditions within an if statement in a Bash script. These operators allow combining expressions to form complex conditional logic.

**Step 1:** In this Creating a file named andnor.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch andnor.sh
santhosh@ccb228d15332571:~$ nano andnor.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.

Script Description:

- &&: Logical AND. All conditions joined by && must be true for the overall condition to be true.
- ||: Logical OR. At least one condition joined by || must be true for the overall condition to be true.

**Condition 1**:

- `[[ 10 -eq 10 && 5 -gt 4 || 3 -eq 4 || 3 -lt 6 ]]`
  - Breakdown:
    - `10 -eq 10`: **True**
    - `5 -gt 4`: **True**
    - `3 -eq 4`: **False**
    - `3 -lt 6`: **True**
  - Evaluation:
    - `TRUE && TRUE`: **True**
    - `True || False`: **True**
    - `True || True`: **True**
  - Result: Outputs `"Condition is true."`

**Condition 2**:

- `[[ 8 -eq 8 && 8 -gt 10 || 9 -lt 5 ]]`
  - Breakdown:
    - `8 -eq 8`: **True**
    - `8 -gt 10`: **False**

- `9 -lt 5`: **False**
- Evaluation:
  - `TRUE && FALSE`: **False**
  - `False || False`: **False**
- Result: Does not output anything (the `if` condition evaluates to false).


**Step 3:** Providing the necessary permissions to the andnor.sh file

```
santhosh@ccb228d15332571:~$ chmod +x andnor.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./andnor.sh
Condition is true.
```

# Example 7

**In this example, we will find "if a given number is greater than 50 and if it is an even number" by using nested if expression.**

**Purpose:** You can apply as many 'if statements' as required inside your bash script. It is also possible to use an if statement inside another 'if statement'. It is known as Nested If Statement.

**Step 1:** In this Creating a file named nested_if.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch nested_if.sh
santhosh@ccb228d15332571:~$ nano nested_if.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.

**Script Description:**

- The script takes a single input argument $1 representing the number to be checked.
- if [ $1 -gt 50 ]: Checks if the number is greater than 50.
- If true, prints "Number is greater than 50.
- if (( $1 % 2 == 0 )): Checks if the number is divisible by 2 (i.e., even).
- If true, prints "and it is an even number."
- If both conditions are met, it displays.
- If only the first condition is met, only the first message is printed.

**Step 3:** Providing the necessary permissions to the nested_if.sh file.

```
santhosh@ccb228d15332571:~$ chmod +x nested_if.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./nested_if.sh 101
Number is greater than 50.
```

# Bash If Else

Bash if-else statements are used to perform conditional tasks in the sequential flow of execution of statements. Sometimes, we want to process a specific set of statements if a condition is true, and another set of statements if it is false. To perform such type of actions, we can apply the if-else mechanism. We can apply the condition with the 'if statement'.

### Important Points:

1. We can use a set of one or more conditions joined using conditional operators.

2. Else block commands includes a set of actions to perform when the condition is false.

3. The semi-colon (;) after the conditional expression is a must.

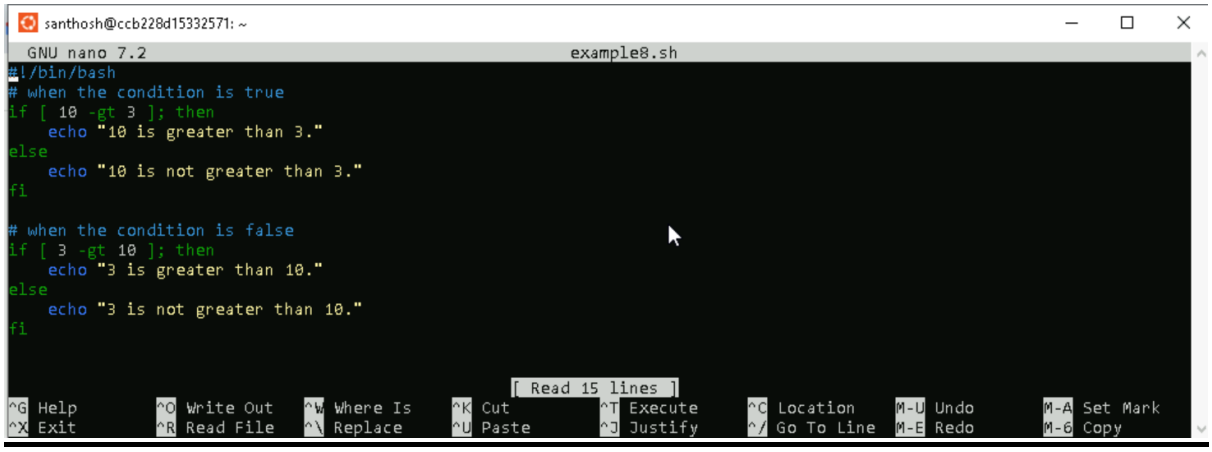
## Example 8

**Purpose:** Following example consists of two different scenarios where in the first if-else statement, the condition is true, and in the second if-else statement, the condition is false.

**Step 1:** In this Creating a file named example8.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch example8.sh
santhosh@ccb228d15332571:~$ nano example8.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.



**Step 3:** Providing the necessary permissions to the example8.sh file

```
santhosh@ccb228d15332571:~$ chmod +x example8.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./example8.sh
10 is greater than 3.
3 is not greater than 10.
```

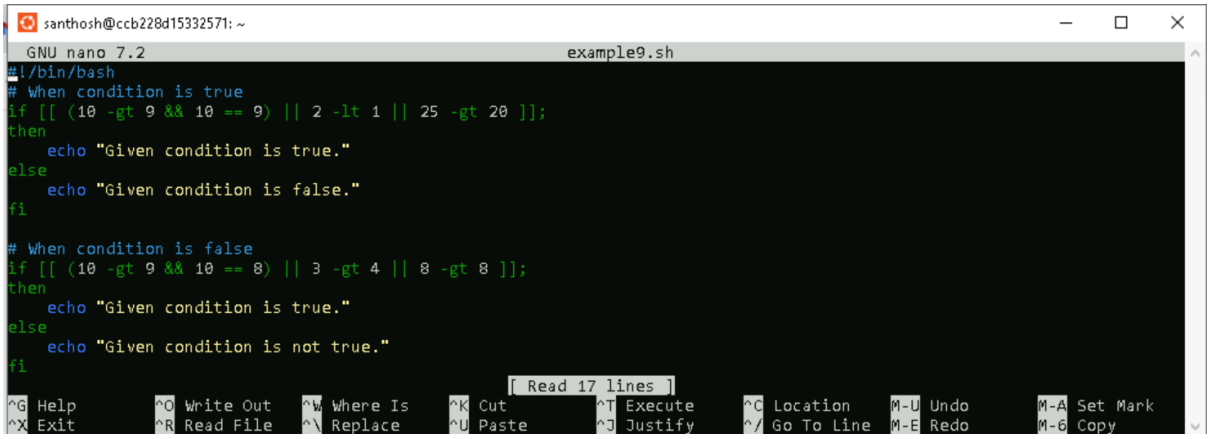
# Example 9

**Purpose:** In this example, we explained how to use multiple conditions with the if-else statement in Bash. We use bash logical operators to join multiple conditions.

**Step 1:** In this Creating a file named example9.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch example9.sh
santhosh@ccb228d15332571:~$ nano example9.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the example9.sh file

```
santhosh@ccb228d15332571:~$ chmod +x example9.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./example9.sh
Given condition is true.
Given condition is not true.
```

# Example 10

**Purpose:** Bash If Else Statement in a Single Line.

We can write complete 'if-else statement' along with the commands in a single line. You need to follow the given rules to use if-else statement in a single line:

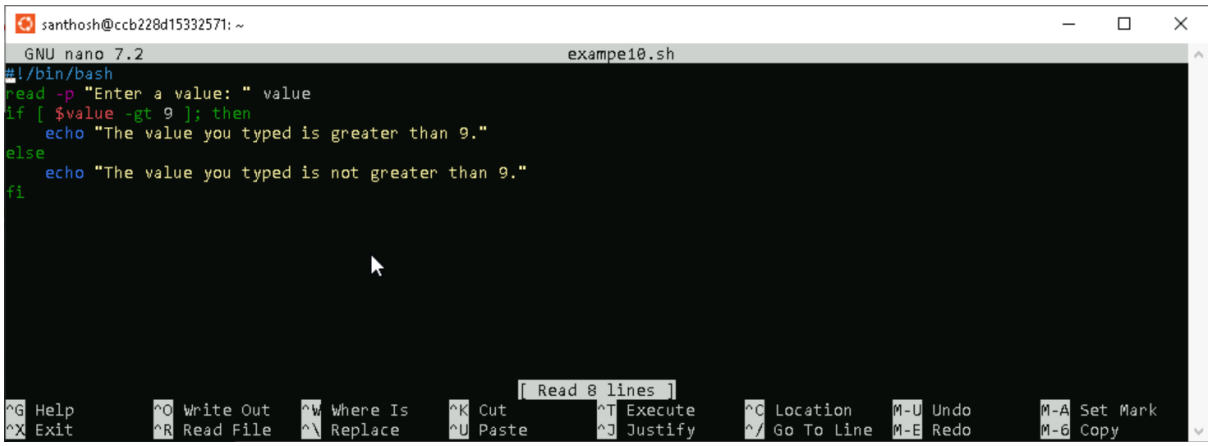
o Use a semi-colon (;) at the end of statements in if and else blocks.

o Use spaces as a delimiter to append all the statements.

**Step 1:** In this Creating a file named exampe10.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch example10.sh
santhosh@ccb228d15332571:~$ nano exampe10.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the exampe10.sh file

```
santhosh@ccb228d15332571:~$ chmod +x exampe10.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./exampe10.sh
Enter a value: 25
The value you typed is greater than 9.
```

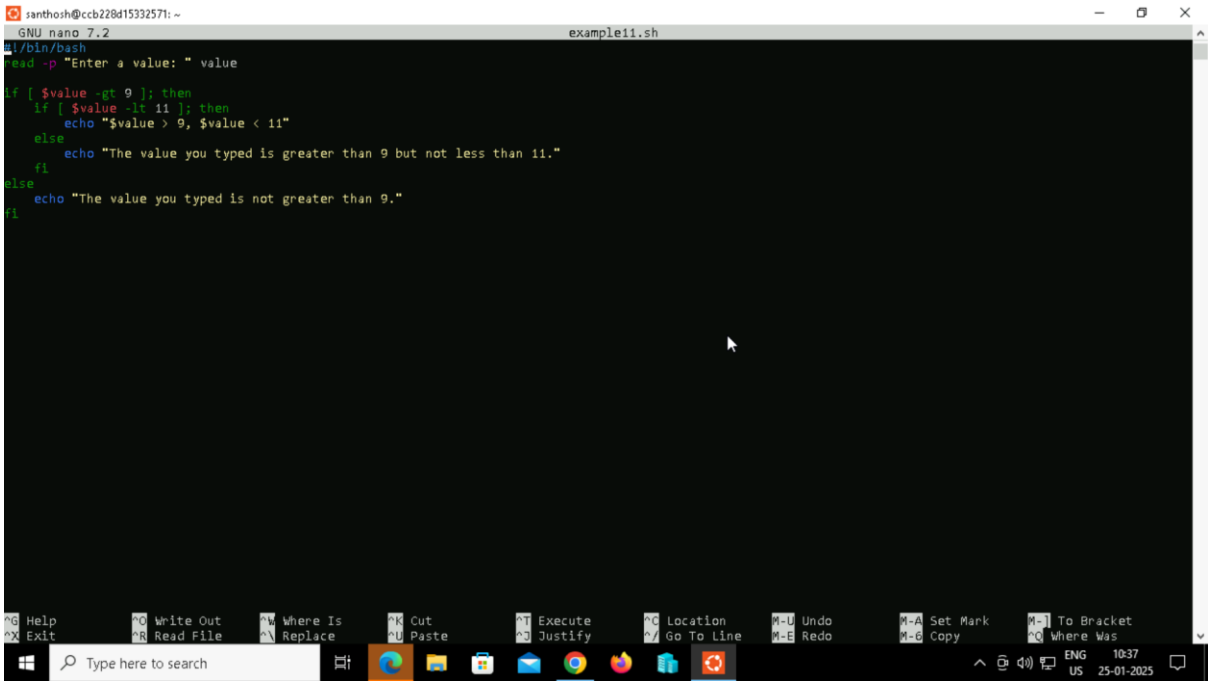
# Example 11

**Purpose:** Bash Nested If Else Just like nested if statement, the if-else statement can also be used inside another if-else statement. It is called nested if-else in Bash scripting.

**Step 1:** In this Creating a file named exampe11.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch example11.sh
santhosh@ccb228d15332571:~$ nano example11.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the exampe11.sh file

```
santhosh@ccb228d15332571:~$ chmod +x example11.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./example11.sh
Enter a value: 10
10 > 9, 10 < 11
```

# Example 12

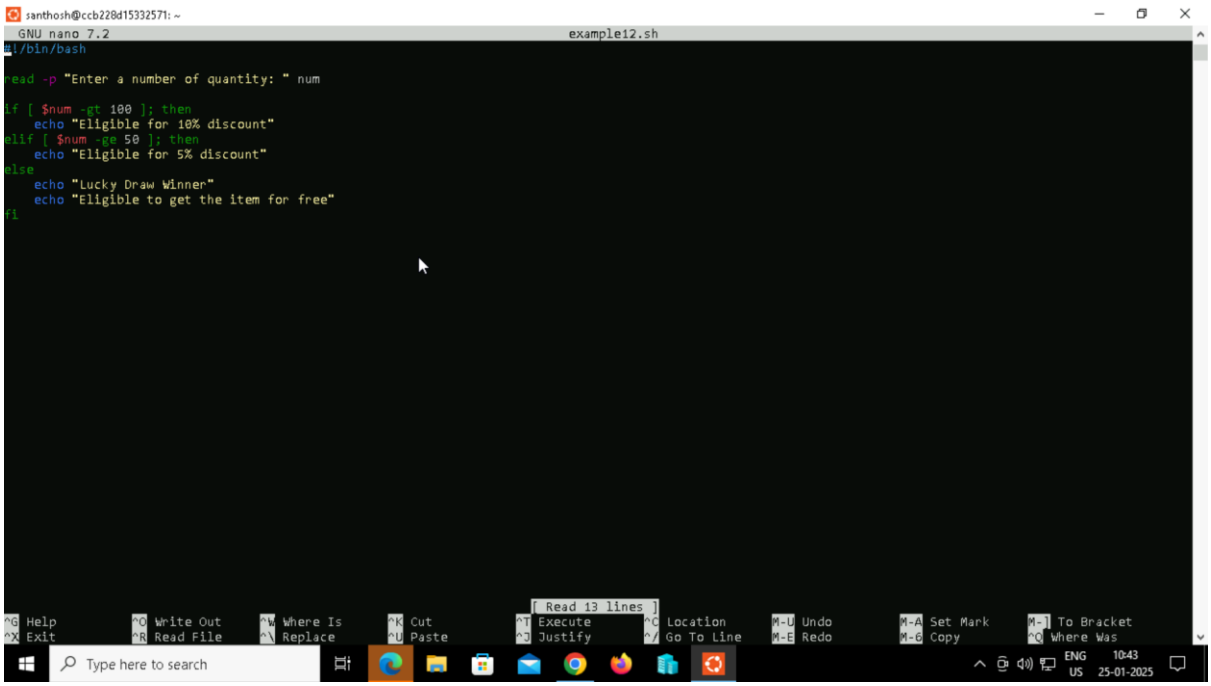
# Bash Else If

Bash else-if statement is used for multiple conditions. It is just like an addition to Bash if-else statement. In Bash elif, there can be several elif blocks with a Boolean expression for each one of them. In the case of the first 'if statement', if a condition goes false, then the second 'if condition' is checked.

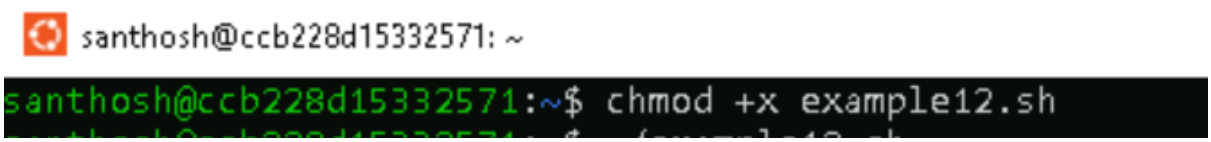
**Step 1:** In this Creating a file named exampe12.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch example12.sh
santhosh@ccb228d15332571:~$ nano example12.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the exampe12.sh file



**Step 4:** Running the and.sh file and getting the output.

a. If we enter the number of quantity as 110, then the condition of 'if statement' evaluates to true and the output looks like:

```
santhosh@ccb228d15332571:~$ ./example12.sh
Enter a number of quantity: 110
Eligible for 10% discount
```

b. If we enter the number of quantity as 90 then condition of 'elif statement' evaluates to true, and the output looks like

```
santhosh@ccb228d15332571:~$ ./example12.sh
Enter a number of quantity: 90
Eligible for 5% discount
```

c. If we enter the number of quantity as 100, then no condition will be true. In this case, the block of commands inside the 'else statement' is executed, and the output looks like:

```
santhosh@ccb228d15332571:~$ ./example12.sh
Enter a number of quantity: 49
Lucky Draw Winner
Eligible to get the item for free
```

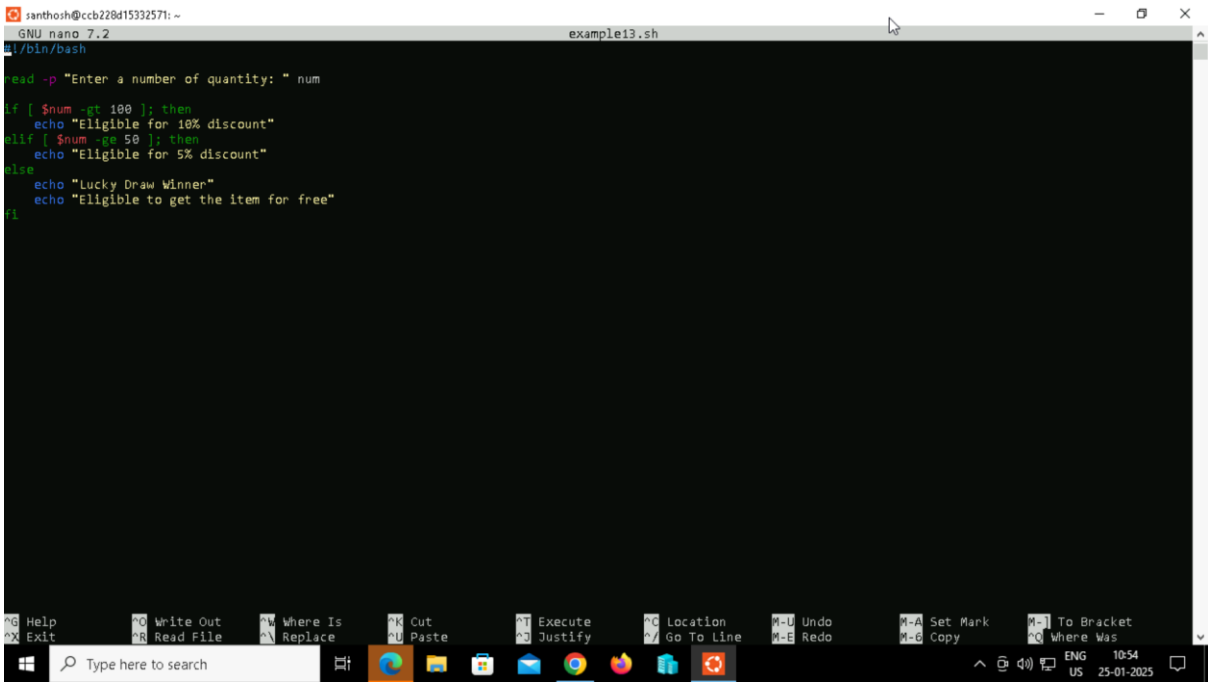
# Example 13

**Purpose:** This example is demonstrating how to use multiple conditions with the else-if statement in Bash. We use bash logical operators to join multiple conditions.

**Step 1:** In this Creating a file named exampe13.sh using touch command and editing the file using the nano command.

```
santhosh@ccb228d15332571:~$ touch example13.sh
santhosh@ccb228d15332571:~$ nano example13.sh
```

**Step 2:** Creating the Shell script comparison in the nano file.




**Step 3:** Providing the necessary permissions to the exampe13.sh file

```
santhosh@ccb228d15332571:~$ chmod +x example13.sh
```

**Step 4:** Running the and.sh file and getting the output.

```
santhosh@ccb228d15332571:~$ ./example13.sh
Enter a number of quantity: 100
Eligible for 5% discount
```