1. Introduction to Computer hardware: Physical identification of major components of a computer system such as mother board, RAM modules, daughter cards, bus slots, SMPS, internal storage devices, interfacing ports.

Procedure:

What is Computer Hardware?

Computer hardware is a hardware part of a computer system. In simple words, only those parts of the computer system which we can see or touch are called computer hardware.

Hardware is an important part of our computer system without which the computer is incomplete. You cannot use a computer without hardware and without hardware, there cannot be a computer system or construction.

1. Mouse

A mouse is a hardware input device that is used to move the cursor or pointer on computer screens. It can also be used to run computer programs, select items in a graphical user interface, and manipulate objects in the computer world. Some common examples of how it can be used are clicking on buttons, scrolling up and down the screen, selecting files, opening folders, and so on.

2. Keyboard

A keyboard is an input device that you use to enter data into a computer. It's also called the input device for your computer. Keyboards are used with PCs, laptops, tablets, and other devices. There are many different types of keyboards, but the most common one is the QWERTY keyboard. A QWERTY keyboard has all the letters in alphabetical order on it. This is different from some other types of keyboards, like Dvorak or Colemak keyboards. For example, these keyboards have keys arranged differently than what you're used to seeing on a QWERTY keyboard. And that means that typing on these keyboards will feel like typing in another language at first! But don't worry - once you get accustomed to it, it feels natural!

3. Monitor

Personal computers use a monitor to display data, run the software, and interact with the user. A monitor is an electronic visual display that connects to your computer or laptop. It is used for displaying images, text, videos, games, web pages, and more. Monitors are available in different sizes depending on the needs of the person using them. The most common types of monitors are CRT (cathode ray tube), LCD (liquid crystal display), and LED (light-emitting diode).

4. Motherboard

The motherboard is the backbone of our computer system. It's the central processing unit or CPU. It connects all the other components, like memory and graphics card, to the power supply. The motherboard is where all the wires are plugged in and it's also where you place your RAM, which is your computer's working memory. The motherboard is what makes one machine different from another.

 Motherboards are made up of tiny transistors that control the flow of electricity through copper tracks on their surface. These transistors are called Integrated Circuits or ICs for short.

5. CPU (Central Processing Unit)

A CPU, or central processing unit, is the brain of a computer. The CPU processes information and runs programs. It functions as a control unit that executes programs according to instructions in its program memory. The CPU contains elements such as registers, an arithmetic logic unit (ALU), and control logic for sequencing instructions.

6. RAM Memory

A computer's RAM is a type of computer memory that stores information so the CPU can access it directly. Computer systems use main memory to store both data and programs. The more RAM you have, the more data your system can process at one time. This will lead to more efficient operations on your computer, which translates into better performance for the user.

7. ROM Memory

ROM stands for a type of memory chip that can be read from but not written to. In other words, it's a form of data storage that can't be changed after being programmed. It's sometimes called "non-volatile" memory because the stored information will remain even when not powered up or in use. ROM is often used to store a computer's basic start-up instructions and certain types of data, such as your car's on-board computer system and a calculator's data tables.

8. Hard Disk Drive

A hard disk drive is a piece of hardware inside a computer that stores information. It's used to store software and data in a safe place, which can be accessed when needed. With magnetic storage, there are no moving parts - unlike a CD or DVD player in which you need to move a disk in order to access data. You can think of it as "a closet" where all your stuff is stored safely. As long as you have power, you can get to your things when you need them.

9. Optical Drive

Optical Drives are used in PCs to read and write CDs and DVDs. The optical drive reads the data from the disc, which can then be transformed into a digital file that is readable by the computer. This makes it easy to backup files, play music or movies, or copy data from one disc to another.
. The term "CD" refers to Compact Discs, which are the most common type of optical drive on modern computers. They are often used for installing software on your computer, moving data between computers, or writing new programs.

shutterstock.com · 117172483

10. Power Supply

A power supply is an electrical appliance that provides the necessary power to operate a computer. Computers are powered by electricity, and the power supply converts the alternating current (AC) from the electric outlet into direct current (DC). The power supply in a computer can be an internal or external component.

2. Install latest version of Ubuntu on a virtualbox
**Procedure:**
1. Download and Virtualbox Windows 10 Installation
2. Ubuntu ISO download
3. Install Virtualbox
4. Create an Ubuntu VM
5. Install Ubuntu on Virtualbox Windows 10
6. Install Virtualbox Guest Additions
Download and Virtualbox Windows 10 Installation
1. Install Ubuntu on VirtualBox
2. HowTo Install Ubuntu On VirtualBox?
2.1. Open VirtualBox
2.2. Click on "New" to create a virtual machine
2.3. Enter Name for your Virtual Machine
2.4. Select "Linux" Operating System from"Type"
 2.5. Click "Next"
2.6. Enter amount of memory (RAM) =1024 MBand click "Next"
2.7. Click "Create" to create hard drive
2.8. Click "Next"
2.9. Click "Next"
2.10. Enter Size of Virtual Hard Drive= 20 GBand Click "Create"
2.11. Select Virtual Machine
2.12. Click on "Start" to start the virtual machine
2.13. Select disk file source
2.14 Afters electing the OS file to be installed click "Open"
2.15. Click "Start"
2.16. Click "Ok"
2.17. Click "Install Ubuntu"
2.18. Click "Continue"
2.19. Click "Install Now"
2.20. Click "Continue"
2.21. Select location and click "Continue"
2.22. Select keyboard layout & click "Continue"
2.23. Fill all the details and Click "Continue"
2.24. Now the installation process will start and installation window will appear
2.25. Click "Restart Now"
2.26. When the system will get restarted the following message will appear. Press "Enter"
2.27. Close the pop-up messages by clicking on the Close (×) button
3. Steps To Maximize The Size Of Ubuntu Desktop
3.1. Go to "Devices"
3.2. Click "Insert Guest Additions CD Image…"
3.3. Click "Run"
3.4. Click "Authenticate"
3.5. Press "Enter"
3.6. Now "Restart" your system for the changes to be applied.
3.7. After the system gets restarted. Go to "View"
3.8. Click "Switch to Full screen"
3.9. Click "Switch"

Output Screenshots:
STEP 1: Installing Virtual Box.

a. Starting pop-up window for the installation

b. Custom setup window to select the features you want and select installation location.
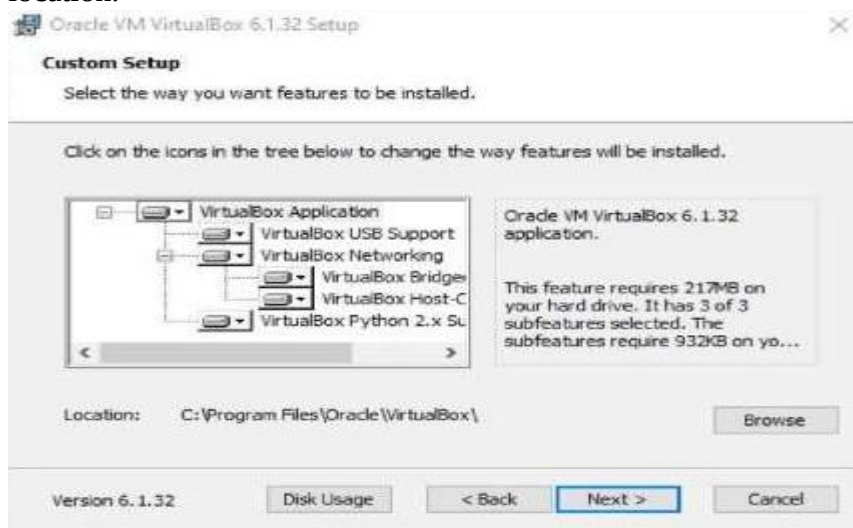
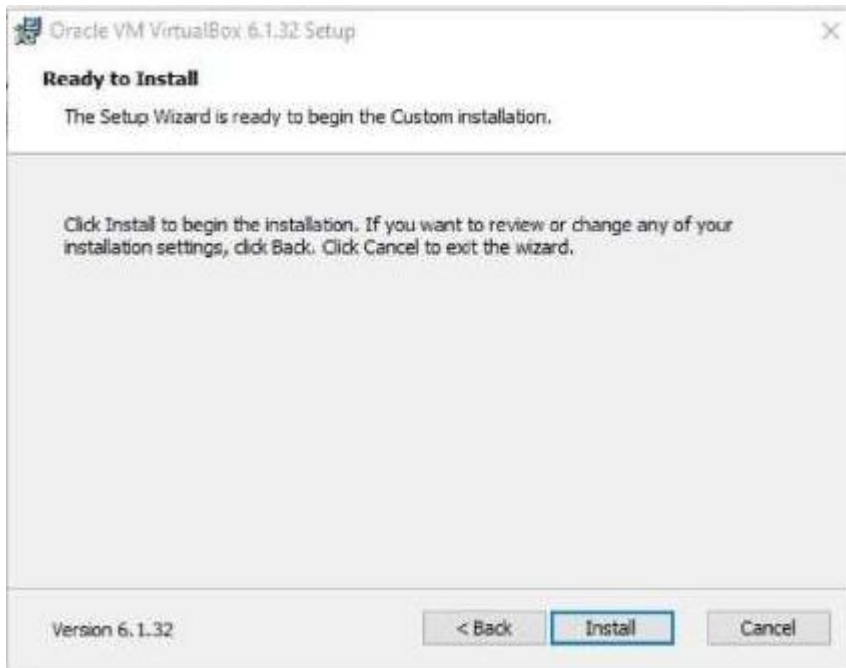**c.** Custom setup window to choose from the option below.
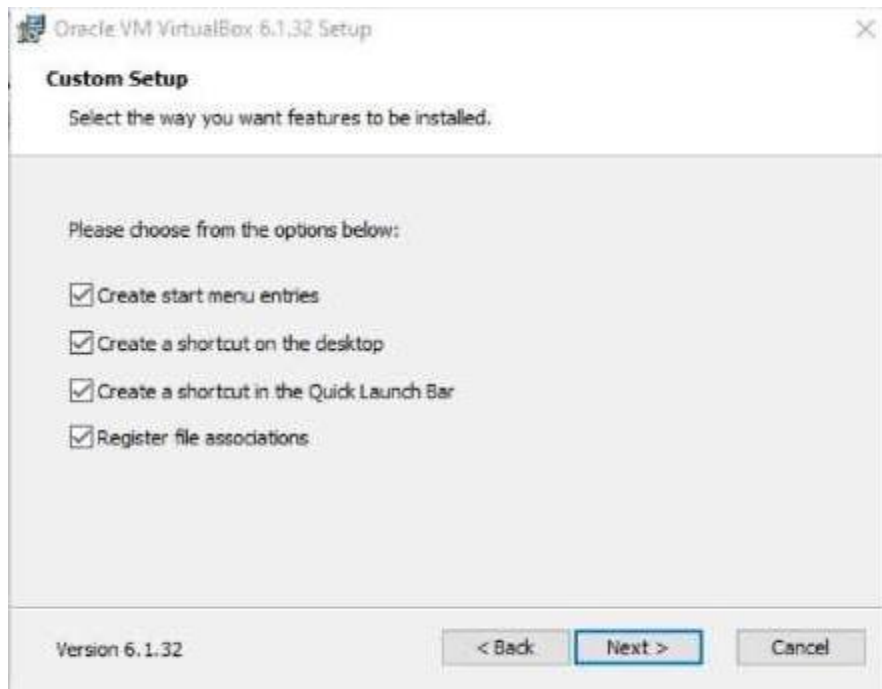
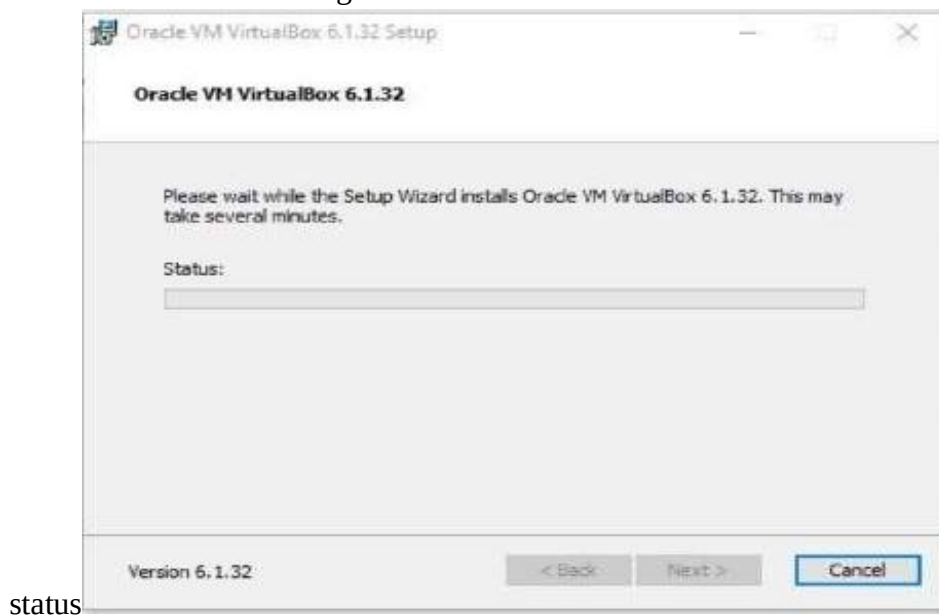**d**. Custom setup window to confirm the installation.

e. Custom setup window to install the virtual box with the install button.
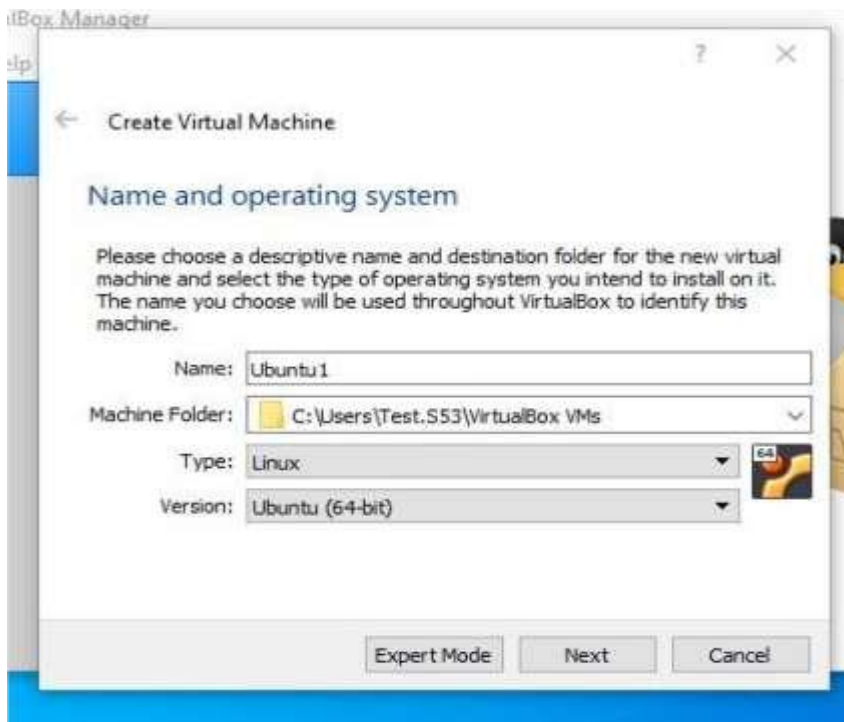

f. Installation box showing the installation

status

g. Installation complete pop-up windows with the finish
button.

STEP 2: Setup the Ubuntu Instance in the VM Virtual Box.

a. After selecting the NEW button to create the Ubuntu instance, the pop-up window to enter & select the name, type and version of the OS.

b. Choose the main memory size for the OS.
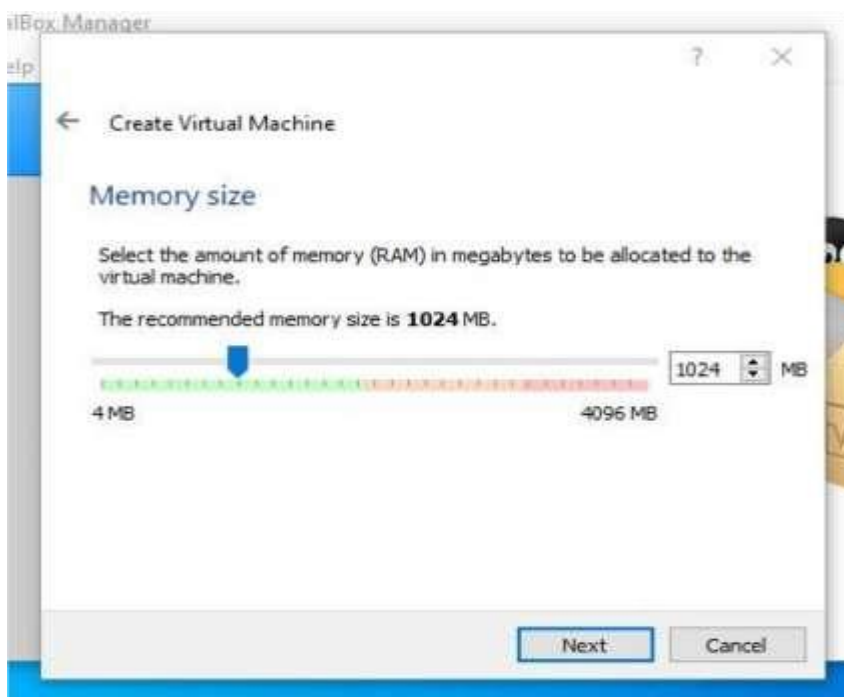
c. Option to add a virtual hard disk to the new machine
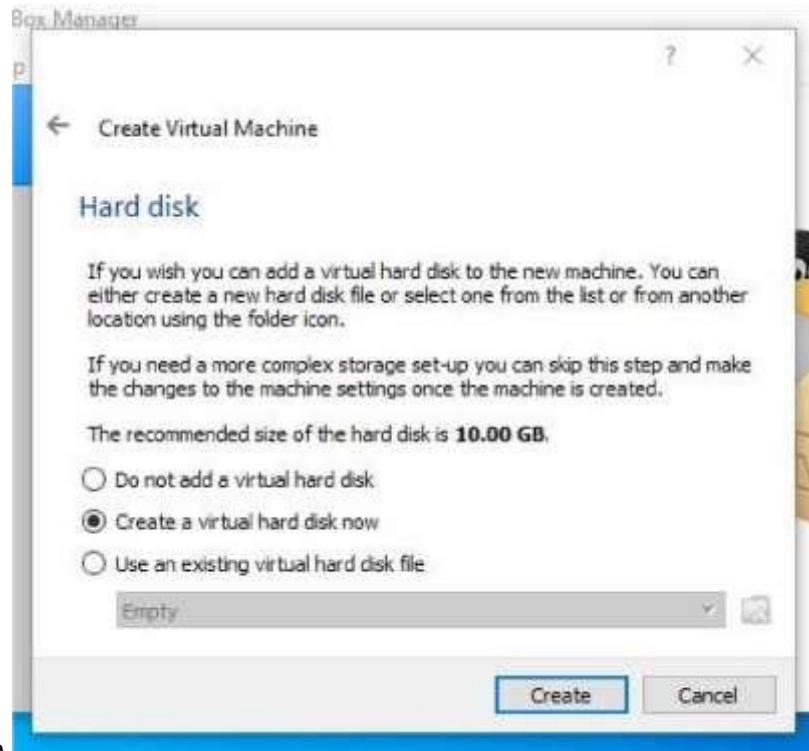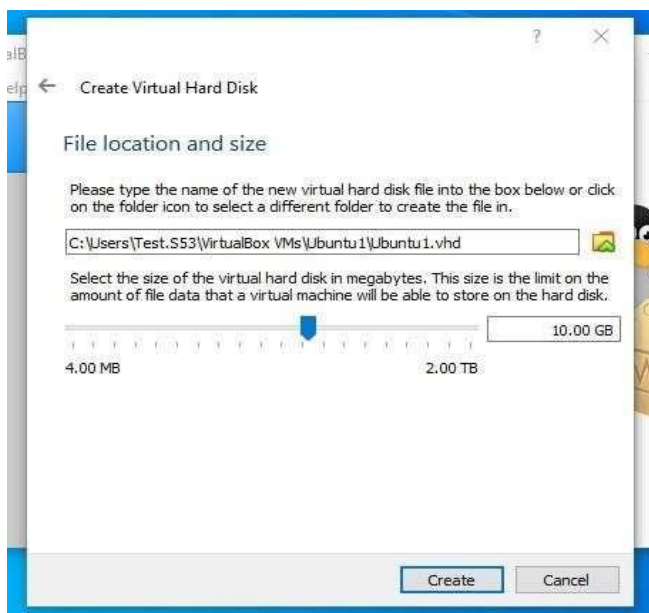

instance.

d. Option to choose the type of new virtual hard disk for new instance of
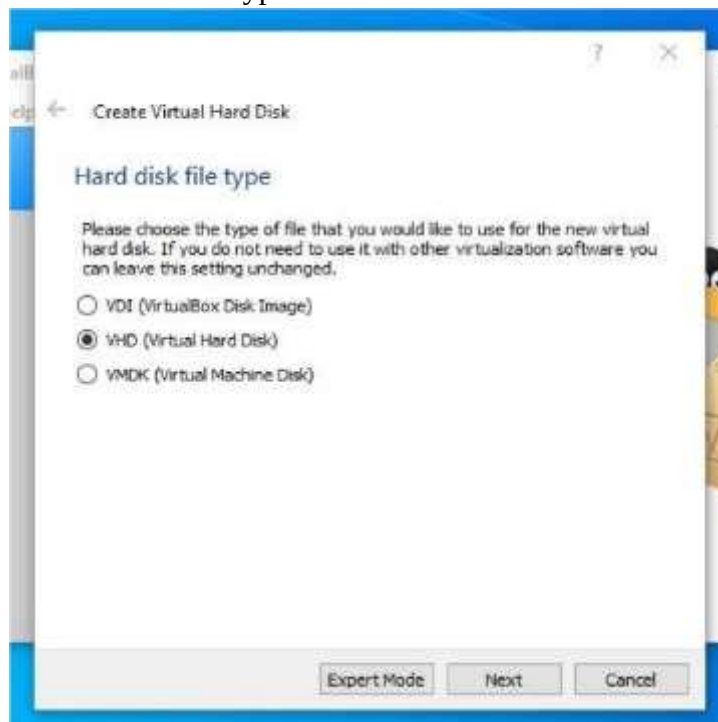
OS.

e. Options to choose the methods of accessing the physical hard disk space for the new instance from the existing hard disk.


f. Panel to select the size of the virtual disk in megabytes and location and name of the

instance and final submit to create the instance of OS.

**g.** The newly created OS instance and at the left side of the application.

h. Settings the configurations of the instance created and adding the ISO image file of the OS correspondingly and selecting the ISO image file from the local device.
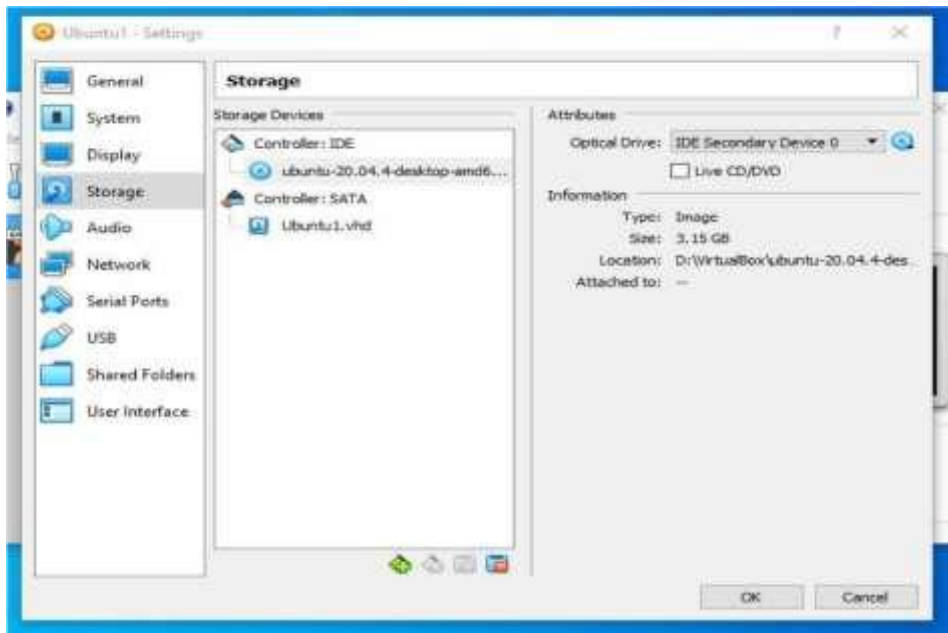
STEP 3: Installation of the Ubuntu OS within the newly created instance.

a. Running the new OS instance and selecting the "Install Ubuntu" to install the loaded ISO file.
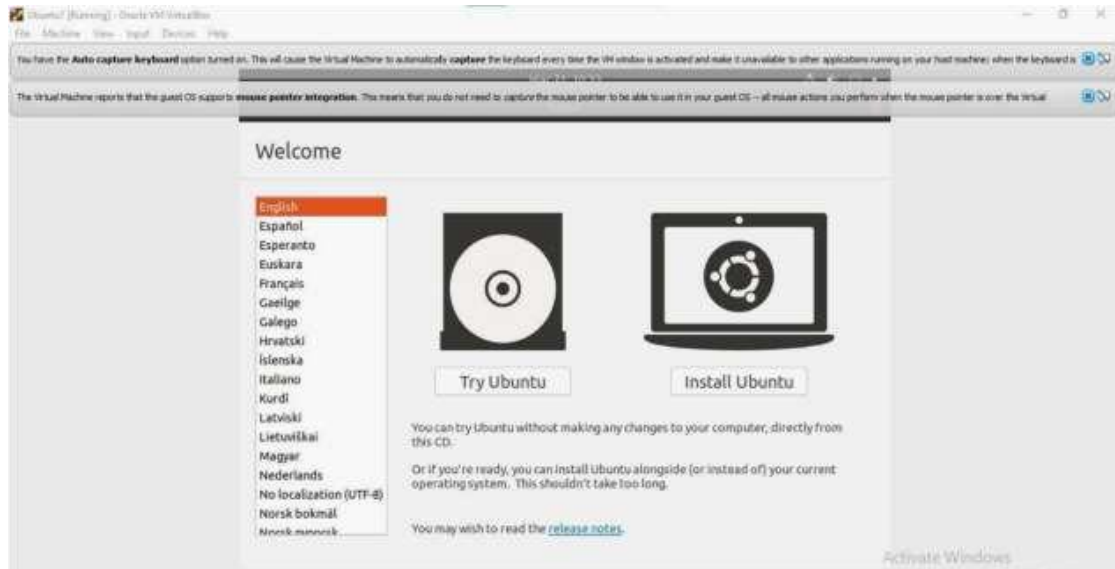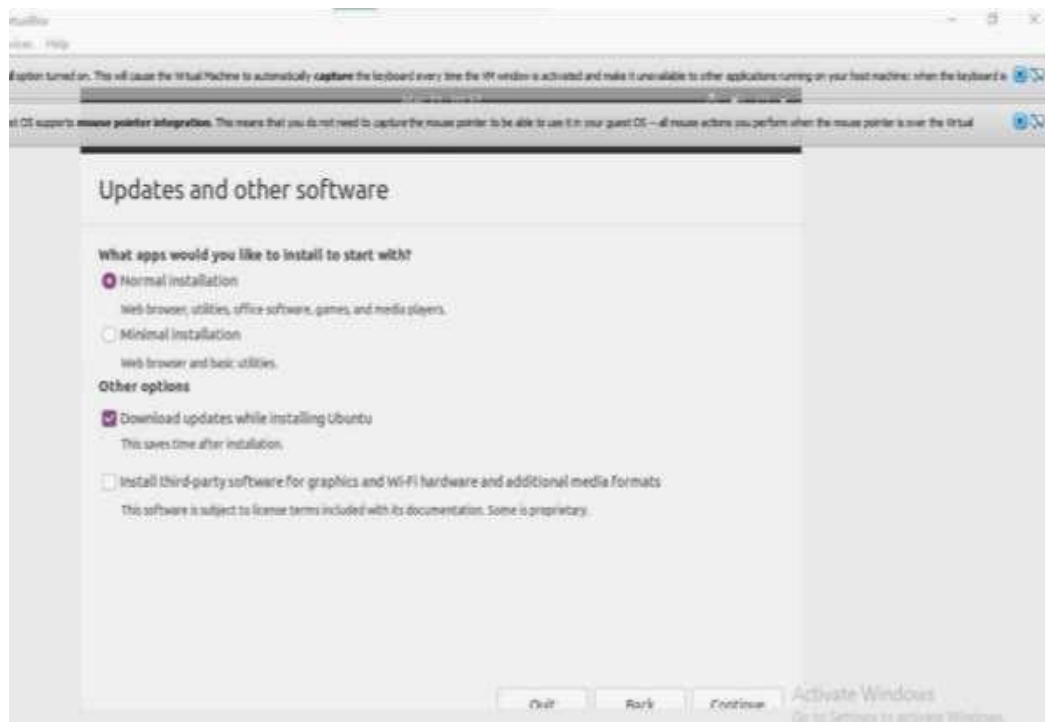

b.   Selecting the language for install the ubuntu OS.

c. Selection of other installation along & within with the installation of ubuntu.

**d.** Selecting the disk partitioning allocation options from the given below.

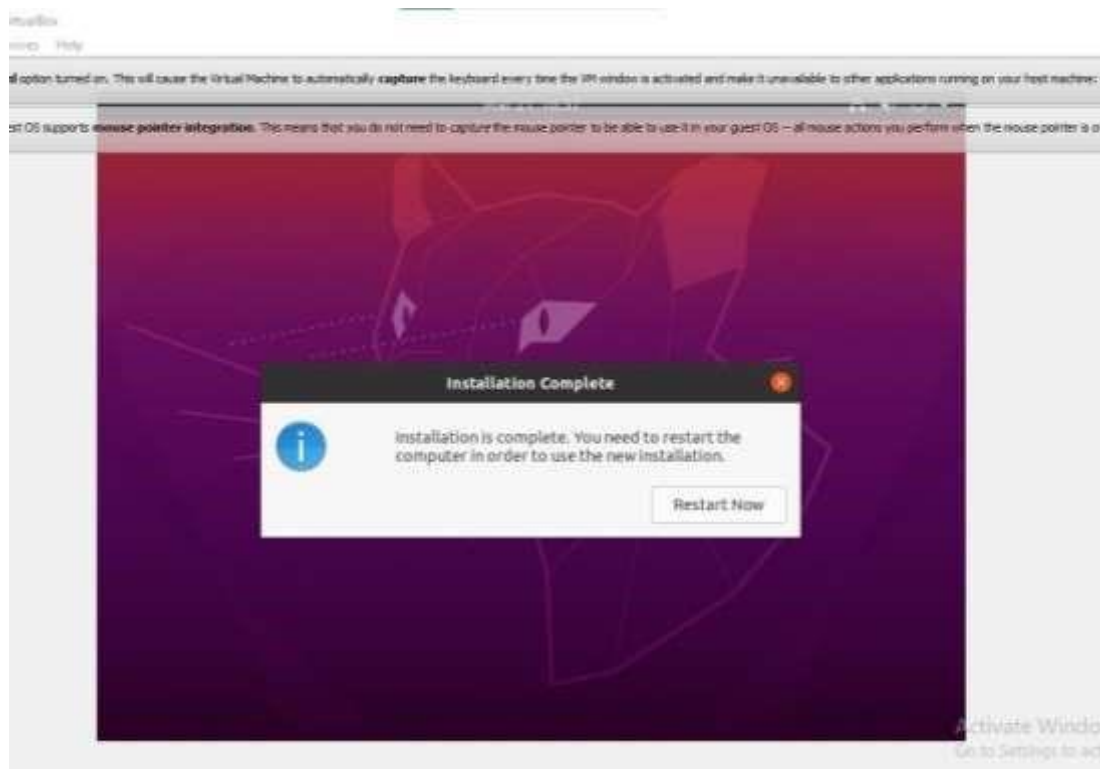**e.** Selecting the geographical location for the time/location.

f. Entering the name, username & password for the account to sign in to the ubuntu OS after installation.
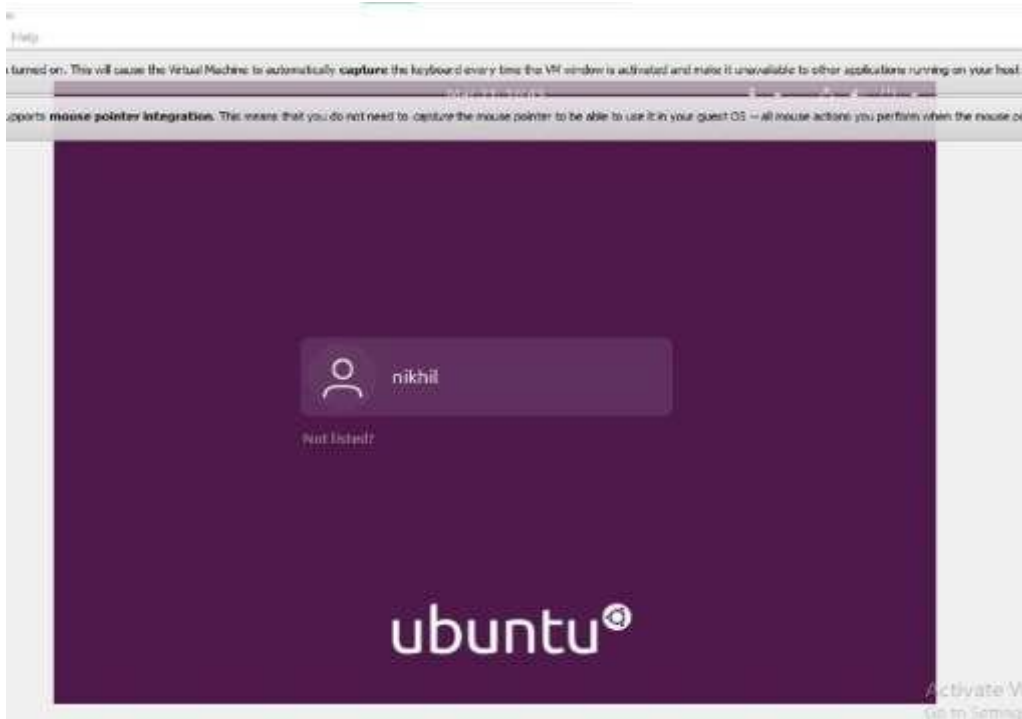
g. Installing the ubuntu OS in the instance, extracting the ubuntu ISO file, setting configurations, setting the various software within, etc.
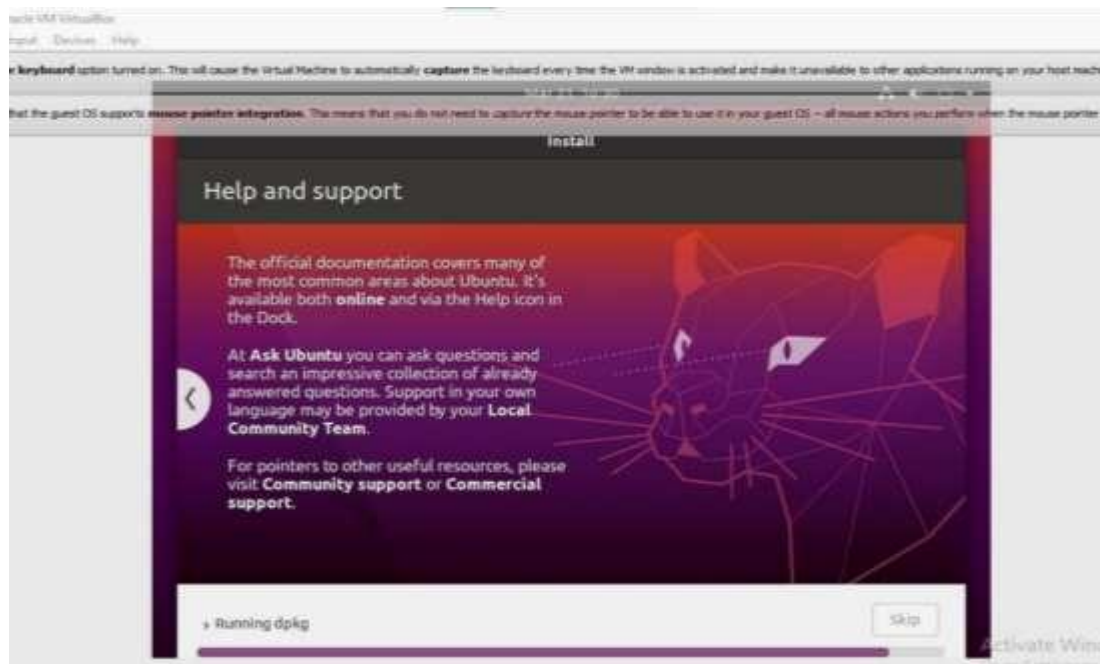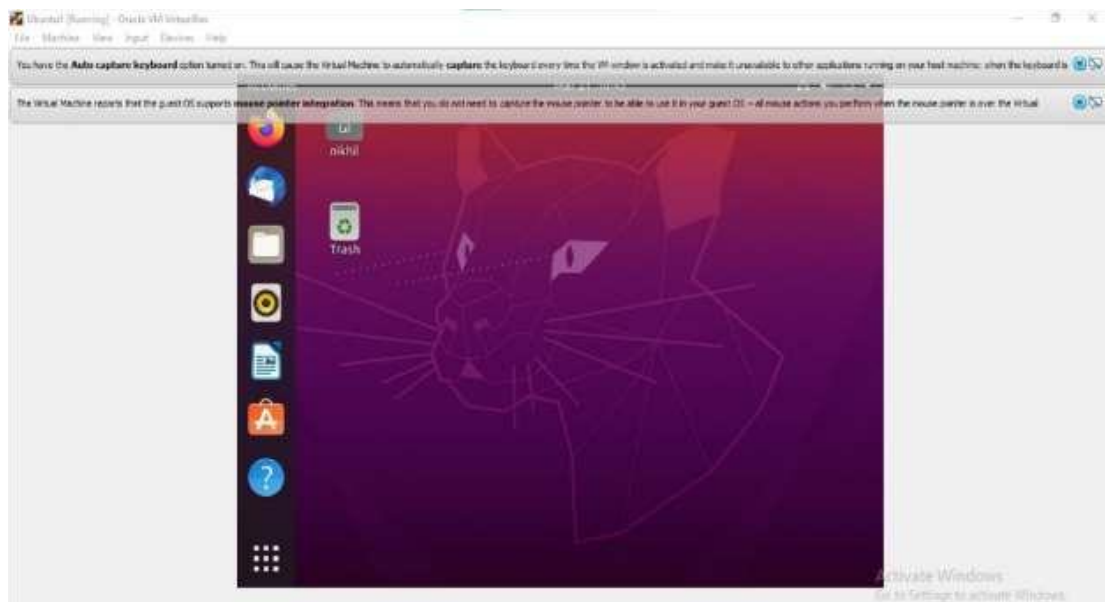

h. Restarting the OS instance to finalize the installation.

i. Signing in and visiting the home screen of the ubuntu OS using the previously registered username & password.

Output:

3. Study of a terminal based text editor such as Vim or Gedit, Basic Linux commands: - familiarity with following commands/operations expected

Procedure

Pwd: This command is used to display the location of the current working directory.

Syntax:-$ pwd

Output:

Mkdir: This command is used to create a new directory under any directory.

Syntax:-$ mkdir<directory name>

Output:

ls: This command is used to display a list of content of directory.

Syntax: -$ ls

Output:

Man: This command is used to display the user manual of any command that we can run on the terminal.

Syntax: -$ man <command name>

Output:

```
student@S47:~$ man pwd
```

Cd: This command is used to change the current directory.

Syntax: -$ cd <directory name>

Output:

```
student@S46:~$ cd stardust
student@S46:~/stardust$ cd ..
```

➢ **cd.. :** This command is used to move to the parent directory of current directory, or the directory one level up from the current directory.

➢ **cd –:** This command is used to switch back to previous directory we were working earlier.

cat > filename: This command is used to view the contents in a file.

Syntax: -$ cat > filename.txt

Output:

```
mca@Z238-UL:~$ cat >a.txt
Hello
```

cat>>filename: This command is used to add contents to an existing file.

Syntax: -$ cat >> filename.txt

Output:

```
mca@Z238-UL:~$ cat >>a.txt
everyone
```

cat filename1 > filename2: This command is used to copy the content from one file to another file.

Syntax: -$ cat filename1 > filename2

Output:

read: This command is used to read the content of a line to a variable.

Syntax: -$ read variable name

Find: This command is used to display contents of particular directory.

Syntax: -$ find filename.txt

grep: This command will let you search through all
the text in a given file.
Syntax: -$ grep word filename.txt

Output:

➢ **grep -i: command** used for a case insensitive search

  Syntax: $ grep -i filename.txt

➢ **grep -v: command** used for inverted search.

  Syntax: $ grep -v filename.txt

➢ **grep -A1:** command used to display line after the result.
  Syntax: $ grep -A1 filename.txt

➢ **grep -B1:** command used to display line before the result.

  Syntax: $ grep -B1 filename.txt

➢ **grep -C1:** command used to display line before and after the result.

  Syntax: $ grep -C1 filename.txt   **wc -word count:** This
  command is used for counting purpose which is used to
  find the number of lines, the number of words, the number
  of characters and the number of bytes.

➢ **wc -l** (count number of lines)

➢ **wc -w** (count number of words)

➢ **wc -c** (count number of characters)

➢ **wc -m** (count number of bytes)

  Syntax: - $ wc -l filename.txt
      $ wc -w filename.txt

        $ wc -c filename.txt

        $ wc -m filename.txt

Output:

df: This command is used to get a report on system disc space usage.

Syntax: -$ df filename.txt

Output:

➢ **df -m: This** command is used to see the report in megabytes.

  Syntax: $ def -m filename.txt

  cut -d: This command is used to cut and display the content based on the delimiter given.

  Syntax: -$ cut –d delimiter –field number filename

```
mca@Z238-UL:~$ cut -d- -f2 b1.txt
98
95
100
mca@Z238-UL:~$ cut -d- -f1 b1.txt
english
maths
malayalam
```

cut -b: This command is used to cut and display the content based on the specified byte number.

Syntax: -$ cut –b byte number filename

Output:

```
mca@Z238-UL:~$ cut -b 2 b1.txt
n
a
a
```

cut --complement -c: This command is used to erase the specified character and display the remaining content of the file.

Syntax: -$ cut --complement –c character number filename.txt

Output:

```
mca@Z238-UL:~$ cut --complement -c 1 b1.txt
nglish  98
aths    95
alayalam        100
```

Paste: This command is used to paste the contents from the specified file.

Syntax: -$ paste filename

Output:

```
mca@Z238-UL:~$ paste football football1
messi     pele
ronaldo maradona
neymar   xavi
mbappe   iniesta
zlatan   puyol
haland
```

More: This command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large.

Syntax: -$ more filename

Cp: This command is used to copy the contents from an existing file to a new file.

Syntax: -$ cpexisting_filenamenew_filename

Output:

```
mca@Z238-UL:~$ ls
Ajay    arraysearch.c  Documents        football1  Pictures  Siddharth
Ajesh   a.txt          Downloads        hello.c    Public    snap
Ajo     b1.txt         examples.desktop Music      R         Templates
a.out   b.txt          folder1          p27.c      sample.c  Videos
Arjun   Desktop        football         pg1.c      Sandra
mca@Z238-UL:~$ cat a.txt
Hello
everyone
mca@Z238-UL:~$ cp a.txt a1.txt
mca@Z238-UL:~$ cat a1.txt
Hello
everyone
```

Mv: This command is used to move an existing file or directory from one location to another.

Syntax: -$ mv filename directory_name

Output:

```
mca@Z238-UL:~$ mv aaa networking
mca@Z238-UL:~$ cd networking
mca@Z238-UL:~/networking$ ls
aaa
```

Head: This command is used to display the first 10 lines of the file by default.
Syntax:-$ head filename

Output:

➢ **head -number**: This command is used to display the lines of the file to the specified number from head.

Tail: This command is used to display the last 10 lines of the file by default.

Syntax: -$ tail filename

Output:

➢ **tail -number**: This command is used to display the lines of the file to the specified number from tail.

sudo useradd: This command is used to add new user.

Syntax: -$ sudo useradd username

Output:

```
mca@Z238-UL:~$ sudo useradd ajay
[sudo] password for mca:
Sorry, try again.
[sudo] password for mca:
```

> **sudo passwd: This** command is used to add password to the user.

  Syntax: -$ sudo passwd username

> **sudo usermod: This** command is used to add members.

  Syntax: -$sudo usermod -G groupname username **delete**

> **sudo userdel username -** used to delete user.
> **sudo groupdel groupname -** used to delete group name.

  Syntax: -$ sudo userdel username

> **sudo groupdel group name chmod: This** command is used change directory permission of files.

  chmod +rwx
  chmod -wx
  chmod +rwx

  Syntax: - $ chmod +rwx filename

  $ chmod -wx filename

  $ chmod -rwx filename

Output:

```
mca@Z238-UL:~$ ls
a1.txt   arraysearch.c   Documents        hello.c     Public      Templates
Ajay     a.txt           Downloads        Music       R           Videos
Ajesh    b1.txt          examples.desktop networking  sample.c
Ajo      bb              folder1          p27.c       Sandra
a.out    b.txt           football         pg1.c       Siddharth
Arjun    Desktop         football1        Pictures    snap
mca@Z238-UL:~$ chmod +rwx a.txt
mca@Z238-UL:~$ chmod -wx a.txt
mca@Z238-UL:~$ cat >>a.txt
bash: a.txt: Permission denied
mca@Z238-UL:~$ chmod -rwx a.txt
mca@Z238-UL:~$ cat a.txt
cat: a.txt: Permission denied
mca@Z238-UL:~$
```

chown: This command is used to give ownership to user.

Syntax: - $ sudo chown username filename

Output:

```
mca@Z238-UL:~$ sudo chwon ajay a.txt
```

Ssh:This command is used to provide a secure encrypted connection between two hosts over an insecure network.

Syntax: - $ ssh mca@ipaddress

Output:

```
mca@Z238-UL:~$ sudo ssh mca@192.168.6.46
ssh: connect to host 192.168.6.46 port 22: Connection timed out
```
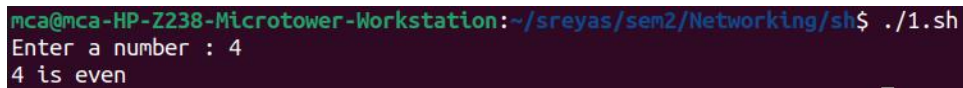
4. SHELL SCRIPTING

1) Write a Shell program to check the given number is even or odd.

program:

```
echo -n "Enter a number : "
read n
if [ `expr $n % 2` -eq 0 ]
then
    echo "$n is even"
else
    echo "$n is odd"
fi
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./1.sh
Enter a number : 4
4 is even
```

2) Write a Shell program to check a leap year.
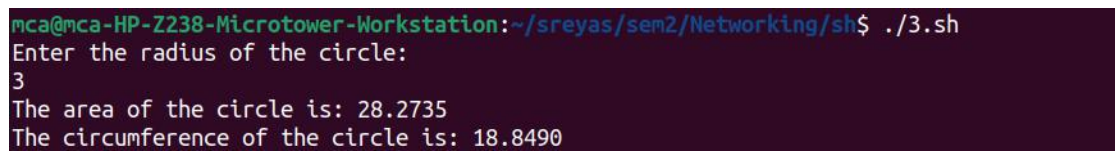
program:

```
echo -n "Enter year : "
read n
if [ `expr $n % 4` -eq 0 ] && [ `expr $n % 100` -ne 0 ] || [ `expr $n % 400` -eq 0 ]; then
    echo "$n is leap year"
else
    echo "$n is not a leap year"
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./2.sh
Enter year : 2024
2024 is leap year
```

3) Write a Shell program to find the area and circumference of a circle.

program:
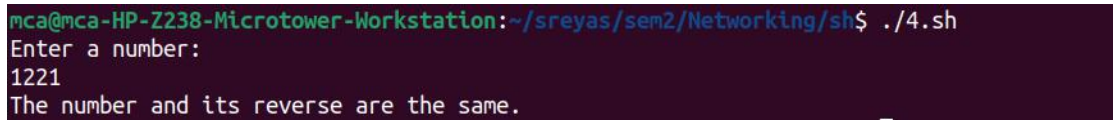
echo "Enter the radius of the circle:"
read radius
area=$(echo "3.1415 * ($radius ^ 2)" | bc)
circumference=$(echo "2 * 3.1415 * $radius" | bc)
echo "The area of the circle is: $area"
echo "The circumference of the circle is: $circumference"

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./3.sh
Enter the radius of the circle:
3
The area of the circle is: 28.2735
The circumference of the circle is: 18.8490
```

4) Write a Shell program to check the given number and its reverse are same.

program:

```
echo "Enter a number:"
read number
reverse=$(echo $number | rev)
if [ $number -eq $reverse ]; then
    echo "The number and its reverse are the same."
else
    echo "The number and its reverse are not the same."
fi
```

output:

5) Write a Shell program to check the given string is palindrome or not.

program:
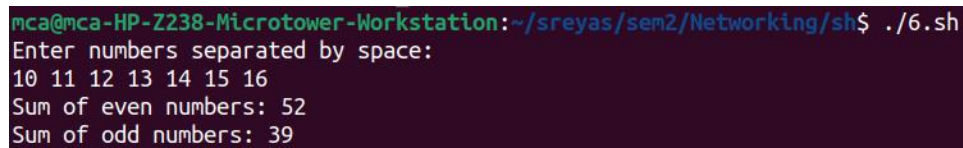
```
echo "Enter a string:"
read string
reverse=$(echo $string | rev)
if [ "$string" == "$reverse" ]; then
    echo "The string is a palindrome."
else
    echo "The string is not a palindrome."
fi
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./5.sh
Enter a string:
hellolleh
The string is a palindrome.
```

6) Write a Shell program to find the sum of odd and even numbers from a set of numbers.

program:
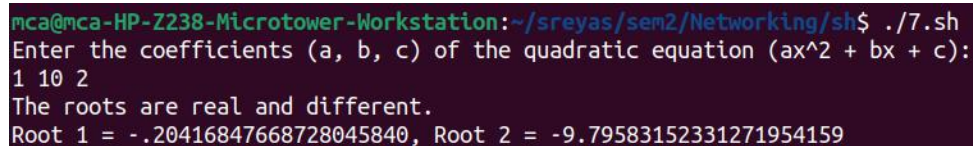
```
echo "Enter numbers separated by space:"
read -a numbers
sum_even=0
sum_odd=0
for num in "${numbers[@]}"; do
   if [ $((num % 2)) -eq 0 ]; then
      sum_even=$((sum_even + num))
   else
      sum_odd=$((sum_odd + num))
   fi
done
echo "Sum of even numbers: $sum_even"
echo "Sum of odd numbers: $sum_odd"
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./6.sh
Enter numbers separated by space:
10 11 12 13 14 15 16
Sum of even numbers: 52
Sum of odd numbers: 39
```

7) Write a Shell program to find the roots of a quadratic equation.

program:

```
echo "Enter the coefficients (a, b, c) of the quadratic equation (ax^2 + bx + c):"
read a b c
discriminant=$((b * b - 4 * a * c))
if [ $discriminant -gt 0 ]; then
    root1=$(echo "(-$b + sqrt($discriminant)) / (2 * $a)" | bc -l)
    root2=$(echo "(-$b - sqrt($discriminant)) / (2 * $a)" | bc -l)
    echo "The roots are real and different."
    echo "Root 1 = $root1, Root 2 = $root2"
elif [ $discriminant -eq 0 ]; then
    root=$(echo "-$b / (2 * $a)" | bc -l)
    echo "The roots are real and equal."
    echo "Root 1 = Root 2 = $root"
else
    real_part=$(echo "-$b / (2 * $a)" | bc -l)
    imaginary_part=$(echo "sqrt($((-1 * discriminant))) / (2 * $a)" | bc -l)
    echo "The roots are complex and different."
    echo "Root 1 = $real_part + $imaginary_part i"
    echo "Root 2 = $real_part - $imaginary_part i"
fi
```

output:

8) Write a Shell program to check the given integer is Armstrong number or not.

program:
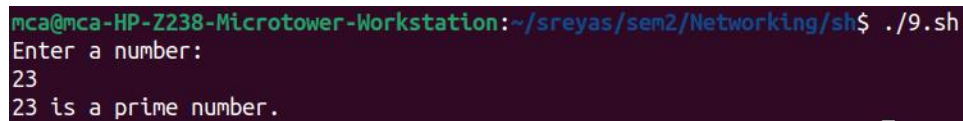
```
echo "Enter a number:"
read number
length=${#number}
sum=0
for ((i=0; i<$length; i++)); do
   digit=${number:i:1}
   sum=$((sum + digit ** length))
done
if [ $sum -eq $number ]; then
   echo "$number is an Armstrong number."
else
   echo "$number is not an Armstrong number."
fi
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./8.sh
Enter a number:
153
153 is an Armstrong number.
```

9) Write a Shell program to check the given integer is prime or not.

program:

```
echo "Enter a number:"
read number
is_prime=true
if [ $number -lt 2 ]; then
   is_prime=false
fi
for ((i=2; i<=number/2; i++)); do
   if [ $((number % i)) -eq 0 ]; then
      is_prime=false
      break
   fi
done
if $is_prime; then
   echo "$number is a prime number."
else
   echo "$number is not a prime number."
fi
```

output:

10) Write a Shell program to generate prime numbers between 1 and 50.

program:

```
echo "Prime numbers between 1 and 50 are:"
for ((i=2; i<=50; i++)); do
   is_prime=true
   for ((j=2; j<=i/2; j++)); do
      if [ $((i % j)) -eq 0 ]; then
         is_prime=false
         break
      fi
   done
   if $is_prime; then
      echo $i
   fi
done
```

output:
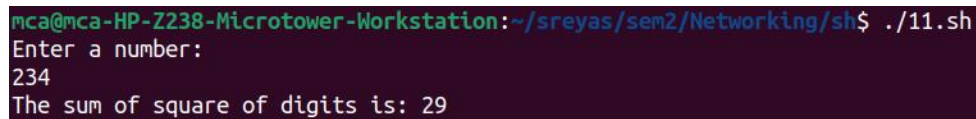
```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./10.sh
Prime numbers between 1 and 50 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

11) Write a Shell program to find the sum of square of individual digits of a number.

program:

```
echo "Enter a number:"
read num
sum=0
while [ $num -gt 0 ]; do
    digit=$(( $num % 10 ))
    sum=$(( $sum + $digit * $digit ))
    num=$(( $num / 10 ))
done
echo "The sum of square of digits is: $sum"
```
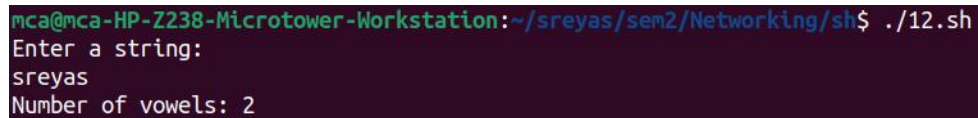
output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./11.sh
Enter a number:
234
The sum of square of digits is: 29
```

12) Write a Shell program to count the number of vowels in a line of text.

program:

echo "Enter a string:"
read str
count=$(echo $str | grep -o -i "[aeiou]" | wc -l)
echo "Number of vowels: $count"

output:

13) Write a Shell program to display student grades.

program:

```
calculate_grade() {
   if [ $1 -ge 90 ]; then
      grade="A"
   elif [ $1 -ge 80 ]; then
      grade="B"
   elif [ $1 -ge 70 ]; then
      grade="C"
   elif [ $1 -ge 60 ]; then
      grade="D"
   else
      grade="F"
   fi
   echo $grade
}

echo "Enter student name:"
read name
echo "Enter student's mark:"
read mark

grade=$(calculate_grade $mark)

echo "Student Name: $name"
echo "Student mark: $mark"
echo "Student Grade: $grade"
```

output:
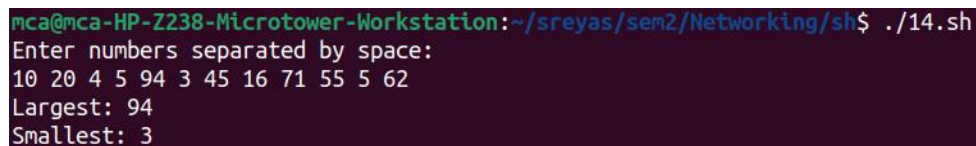
```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./13.sh
Enter student name:
sreyas
Enter student's mark:
90
Student Name: sreyas
Student mark: 90
Student Grade: A
```

14) Write a Shell program to find the smallest and largest numbers from a set of numbers.

program:

```
echo "Enter numbers separated by space:"
read -a numbers
largest=${numbers[0]}
smallest=${numbers[0]}
for num in "${numbers[@]}"; do
    if [ $num -gt $largest ]; then
        largest=$num
    fi
    if [ $num -lt $smallest ]; then
        smallest=$num
    fi
done
echo "Largest: $largest"
echo "Smallest: $smallest"
```
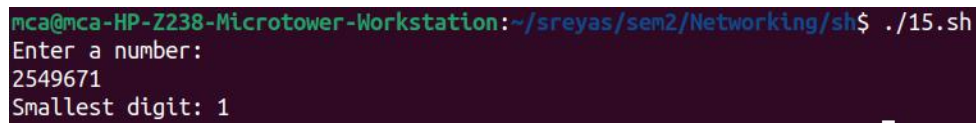
output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./14.sh
Enter numbers separated by space:
10 20 4 5 94 3 45 16 71 55 5 62
Largest: 94
Smallest: 3
```

15) Write a Shell program to find the smallest digit from a number.

program:

echo "Enter a number:"
read num
smallest=$(echo $num | grep -o "[0-9]" | sort | head -n1)
echo "Smallest digit: $smallest"

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./15.sh
Enter a number:
2549671
Smallest digit: 1
```

16) Write a Shell program to find the sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5.

program:

```
sum=0
for ((i=50; i<=100; i++)); do
   if [ $((i % 3)) -eq 0 ] && [ $((i % 5)) -ne 0 ]; then
      sum=$((sum + i))
   fi
done
echo "Sum of numbers divisible by 3 and not by 5 between 50 and 100: $sum"
```

output:
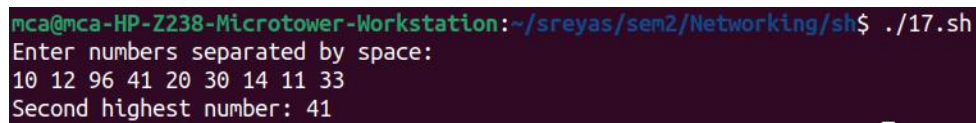
```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./16.sh
Sum of numbers divisible by 3 and not by 5 between 50 and 100: 1050
```

17) Write a Shell program to find the second highest number from a set of numbers.

program:

```
echo "Enter numbers separated by space:"
read -a numbers
IFS=$'\n' sorted=($(sort -n <<<"${numbers[*]}"))
len=${#sorted[@]}
echo "Second highest number: ${sorted[len-2]}"
```
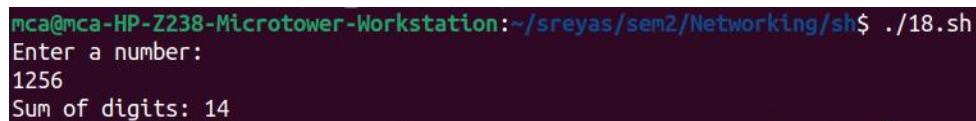
output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./17.sh
Enter numbers separated by space:
10 12 96 41 20 30 14 11 33
Second highest number: 41
```

18) Write a Shell program to find the sum of digits of a number using function.

program:
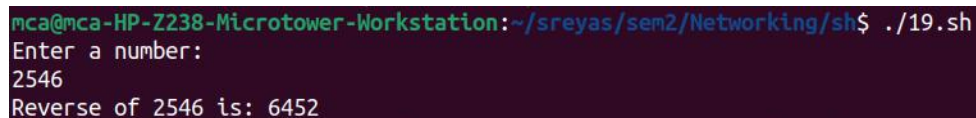
```
echo "Enter a number:"
read num
sum_digits() {
   local n=$1
   local sum=0
   while [ $n -gt 0 ]; do
      sum=$((sum + n % 10))
      n=$((n / 10))
   done
   echo $sum
}
echo "Sum of digits: $(sum_digits $num)"
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./18.sh
Enter a number:
1256
Sum of digits: 14
```

19) Write a Shell program to print the reverse of a number using function.

program:
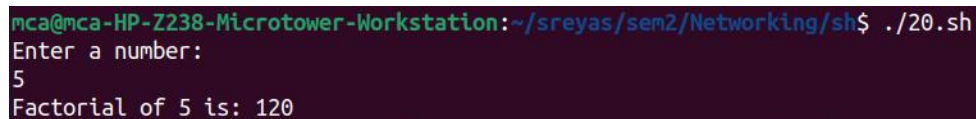
```
echo "Enter a number:"
read num
reverse() {
    local n=$1
    local rev=0
    while [ $n -gt 0 ]; do
        remainder=$((n % 10))
        rev=$((rev * 10 + remainder))
        n=$((n / 10))
    done
    echo $rev
}
echo "Reverse of $num is: $(reverse $num)"
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./19.sh
Enter a number:
2546
Reverse of 2546 is: 6452
```

20) Write a Shell program to find the factorial of a number using for loop.

program:

```
echo "Enter a number:"
read num
fact(){
    fact=1
    for((i=1; i<=num; i++)); do
        fact=$((fact * i))
    done
    echo $fact
}
echo "Factorial of $num is: $(fact)"
```
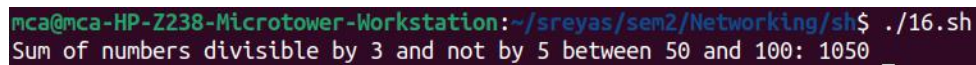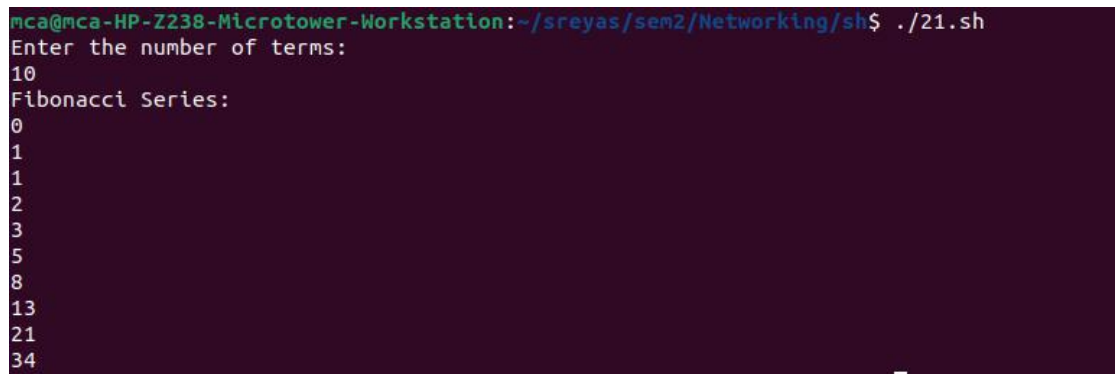
output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./20.sh
Enter a number:
5
Factorial of 5 is: 120
```

21) Write a Shell program to generate Fibonacci series.

program:

```
echo "Enter the number of terms:"
read n
a=0
b=1
echo "Fibonacci Series:"
for ((i=0; i<n; i++)); do
    echo -n "$a "
    fn=$((a + b))
    a=$b
    b=$fn
done
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./21.sh
Enter the number of terms:
10
Fibonacci Series:
0
1
1
2
3
5
8
13
21
34
```

22) Write a shell script, which receives two filenames as arguments. It checks whether the two files contents are same or not. If they are same then second file is deleted.

program:

```
if [ $# -ne 2 ]; then
    echo "Usage: $0 <file1> <file2>"
    exit 1
fi
if cmp -s "$1" "$2"; then
    echo "Files are the same. Deleting $2"
    rm $2
else
    echo "Files are different"
fi
```

output:

23) Write a Menu driven Shell script that Lists current directory, Prints Working Directory, displays Date and displays Users logged in.

program:

```
PS3="Select option: "
select opt in "List current directory" "Print working directory" "Display date" "Display
users logged in" "Exit"; do
    case $opt in
        "List current directory")
            ls
            ;;
        "Print working directory")
            pwd
            ;;
        "Display date")
            date
            ;;
        "Display users logged in")
            who
            ;;
        "Exit")
            break
            ;;
        *)
            echo "Invalid option"
            ;;
    esac
done
```

output:
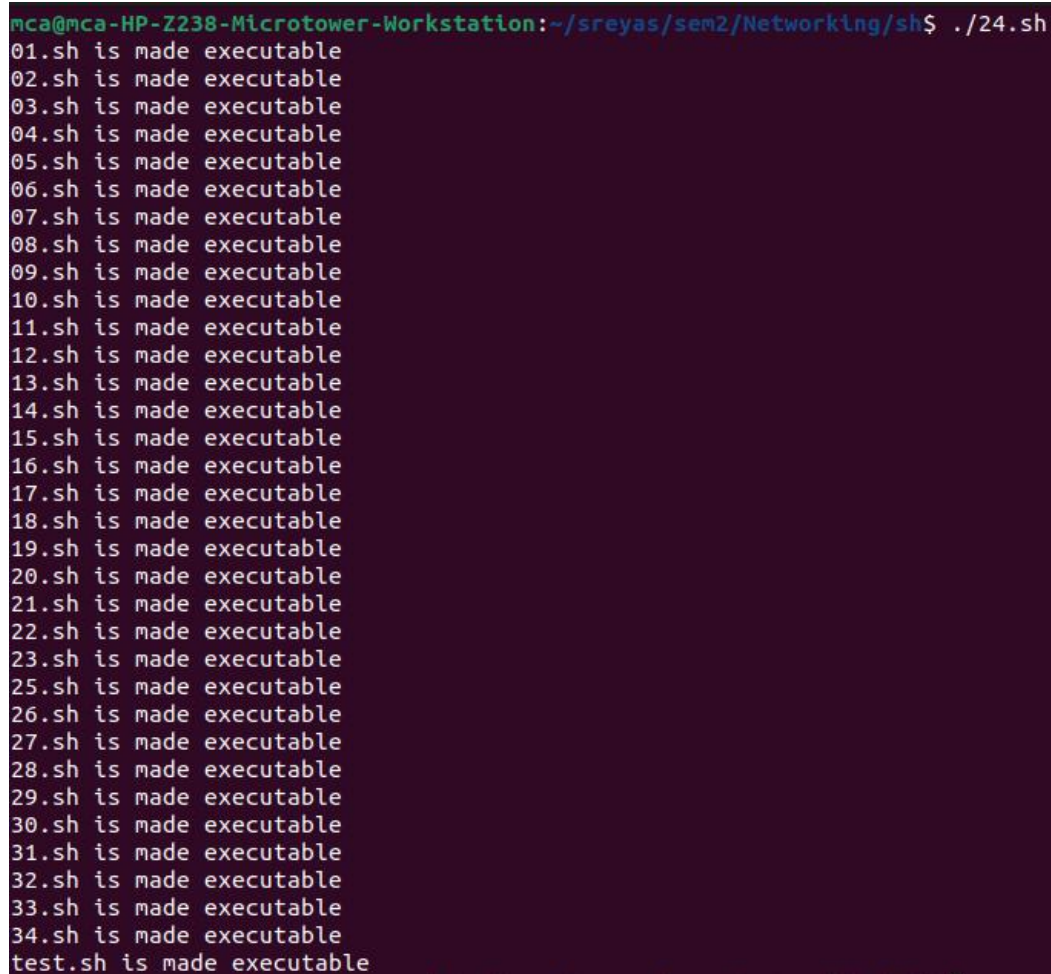
24) Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.

program:

```
for file in *; do
    if [ -f $file ] && [ ! -x $file ]; then
        chmod +x $file
        echo "$file is made executable"
    fi
done
```

output:
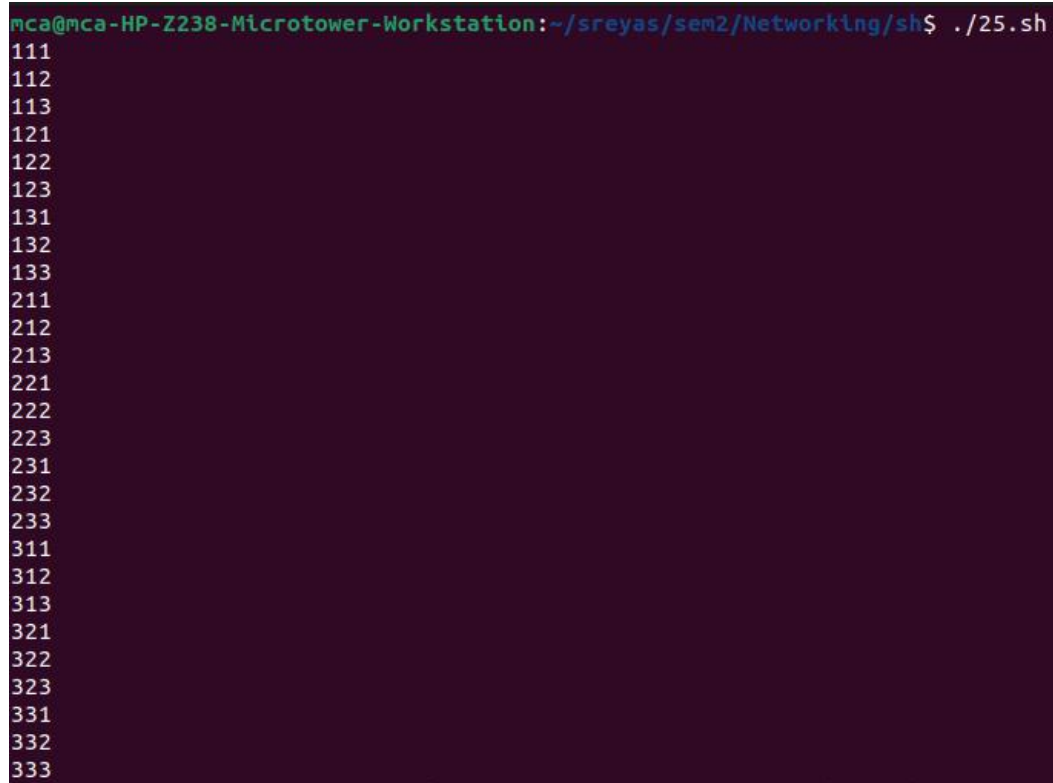
```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./24.sh
01.sh is made executable
02.sh is made executable
03.sh is made executable
04.sh is made executable
05.sh is made executable
06.sh is made executable
07.sh is made executable
08.sh is made executable
09.sh is made executable
10.sh is made executable
11.sh is made executable
12.sh is made executable
13.sh is made executable
14.sh is made executable
15.sh is made executable
16.sh is made executable
17.sh is made executable
18.sh is made executable
19.sh is made executable
20.sh is made executable
21.sh is made executable
22.sh is made executable
23.sh is made executable
25.sh is made executable
26.sh is made executable
27.sh is made executable
28.sh is made executable
29.sh is made executable
30.sh is made executable
31.sh is made executable
32.sh is made executable
33.sh is made executable
34.sh is made executable
test.sh is made executable
```

25) Write a Shell program to generate all combinations of 1, 2, and 3 using loop.

program:

```
for i in 1 2 3; do
   for j in 1 2 3; do
      for k in 1 2 3; do
         echo "$i$j$k"
      done
   done
done
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./25.sh
111
112
113
121
122
123
131
132
133
211
212
213
221
222
223
231
232
233
311
312
313
321
322
323
331
332
333
```

26) Write a Shell program to create the number series.

program:
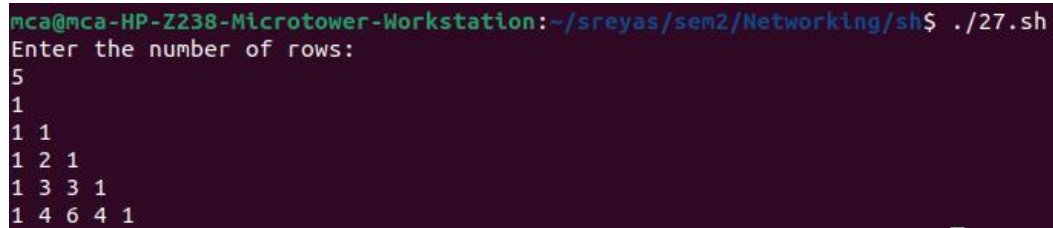
```
echo "Enter the number of terms:"
read n
echo "Number series:"
for ((i=1; i<=n; i++)); do
    echo -n "$i "
done
```
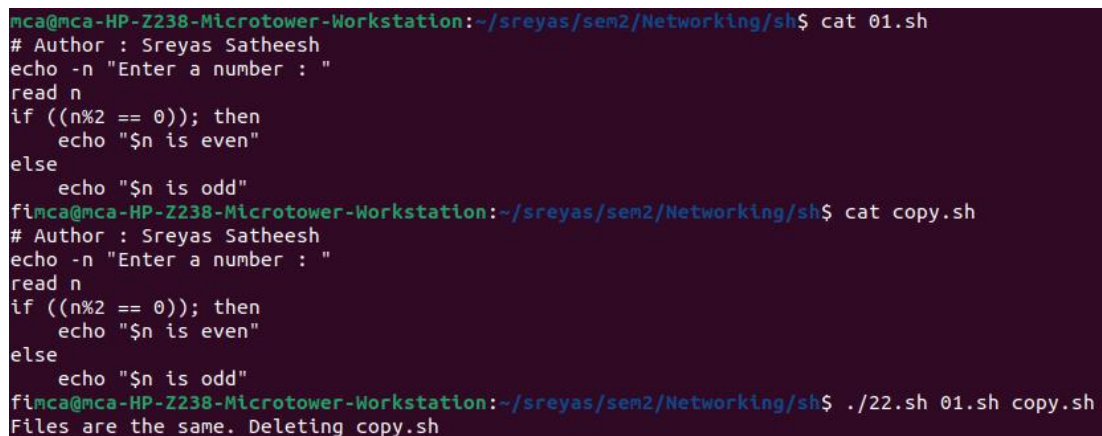
output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./26.sh
Enter the number of terms:
10
Number series:
1 2 3 4 5 6 7 8 9 10
```

27) Write a Shell program to create Pascal's triangle.

program:
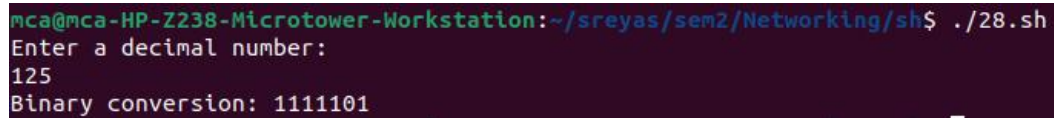
```
echo "Enter the number of rows:"
read rows
for ((i=0; i<rows; i++)); do
   for ((j=0; j<=i; j++)); do
     if [ $j -eq 0 ] || [ $i -eq $j ]; then
        coef=1
     else
        num=$((i-j+1))
        den=$j
        coef=$((coef * num / den))
     fi
     echo -n "$coef "
   done
   echo
done
```

output:

28) Write a Decimal to Binary Conversion Shell Script.

program:

```
echo "Enter a decimal number:"
read decimal
echo "Binary conversion: $(echo "obase=2; $decimal" | bc)"
```
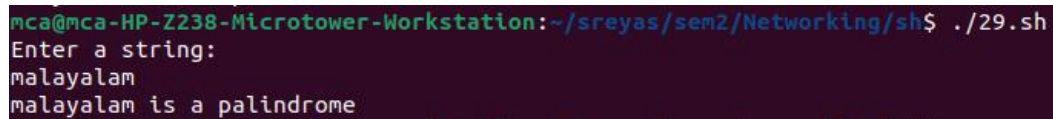
output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./28.sh
Enter a decimal number:
125
Binary conversion: 1111101
```

29) Write a Shell Script to Check Whether a String is Palindrome or not.

program:

```
echo "Enter a string:"
read str
reverse=$(echo $str | rev)
if [ "$str" = "$reverse" ]; then
    echo "$str is a palindrome"
else
    echo "$str is not a palindrome"
fi
```

output:
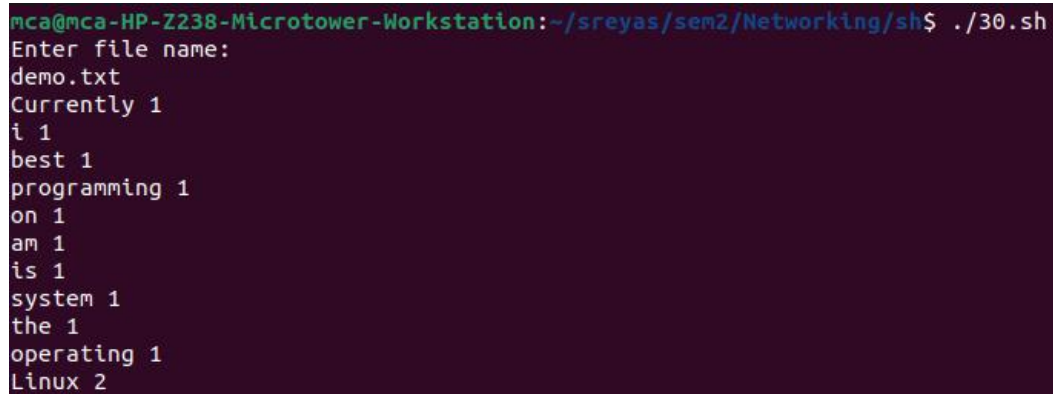

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./29.sh
Enter a string:
malayalam
malayalam is a palindrome
```

30) Write a shell script to find out the unique words in a file and also count the occurrence of each of these words.

program:

```
echo "Enter file name:"
read filename
awk '{for(i=1;i<=NF;i++) a[$i]++} END {for(k in a) print k, a[k]}' $filename
```

output:
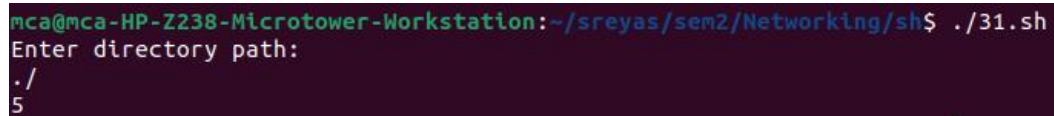
```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./30.sh
Enter file name:
demo.txt
Currently 1
i 1
best 1
programming 1
on 1
am 1
is 1
system 1
the 1
operating 1
Linux 2
```

31) Write a shell script to get the total count of the word "Linux" in all the ".txt" files and also across files present in subdirectories.

program:

```
echo "Enter directory path:"
read dir
grep -roh "Linux" $dir | wc -w
```

output:

32) Write a shell script to validate password strength. Here are a few assumptions for the password string. (Length – minimum of 8 characters. Contain both alphabet and number. Include both the small and capital case letters.)

program:

```
echo "Enter password:"
read password
if [[ ${#password} -lt 8 ]]; then
    echo "Password length should be at least 8 characters"
    exit 1
fi
if ! [[ $password =~ [0-9] ]]; then
    echo "Password should contain at least one digit"
    exit 1
fi
if ! [[ $password =~ [A-Z] ]]; then
    echo "Password should contain at least one uppercase letter"
    exit 1
fi
if ! [[ $password =~ [a-z] ]]; then
    echo "Password should contain at least one lowercase letter"
    exit 1
fi
echo "Password is strong"
```

output:
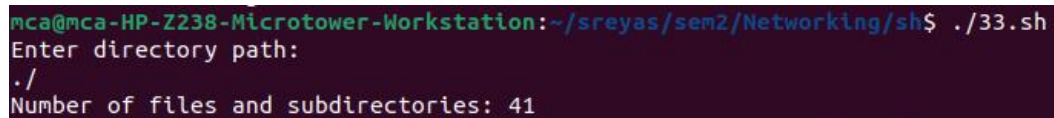
```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./32.sh
Enter password:
Sreyas555
Password is strong
```

33) Write a shell script to print the count of files and subdirectories in the specified directory.

program:

echo "Enter directory path:"
read dir
echo "Number of files and subdirectories: $(find $dir -type d -or -type f | wc -l)"

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./33.sh
Enter directory path:
./
Number of files and subdirectories: 41
```

34) Write a shell script to reverse the list of strings and reverse each string further in the list.
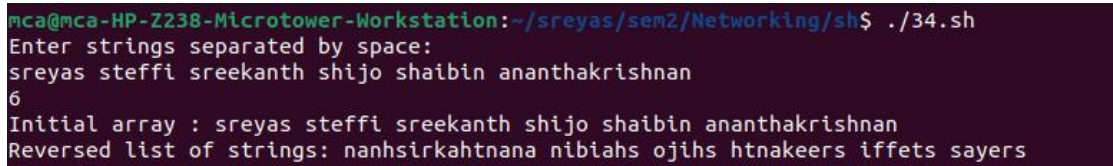
program:

```
echo "Enter strings separated by space:"
read -a strings

leng=${#strings[@]}
echo $leng
reversed_strings=()
for ((i=($leng-1); i>=0; i--)); do
    reversed_strings+=(${strings[i]})
done

echo "Initial array : ${strings[@]}"
echo -n "Reversed list of strings: "
for string in "${reversed_strings[@]}"; do
    echo -n "$(echo $string | rev) "
done; echo
```

output:

```
mca@mca-HP-Z238-Microtower-Workstation:~/sreyas/sem2/Networking/sh$ ./34.sh
Enter strings separated by space:
sreyas steffi sreekanth shijo shaibin ananthakrishnan
6
Initial array : sreyas steffi sreekanth shijo shaibin ananthakrishnan
Reversed list of strings: nanhsirkahtnana nibiahs ojihs htnakeers iffets sayers
```

5. File system hierarchy in a common Linux distribution, file and device permissions, study of system configuration files in /etc., familiarizing log files for system events, user activity, network events.

The Linux File Hierarchy Structure or the Filesystem Hierarchy Standard (FHS) defines the
directory structure and directory contents in Unix-like operating systems. It is maintained by
the Linux Foundation.

In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices.

Some of these directories only exist on a particular system if certain subsystems, such
as the X Window System, are installed.

Most of these directories exist in all UNIX operating systems and are generally used in
much the same way; however, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux.

File System Hierarchy(FHS) of Linux
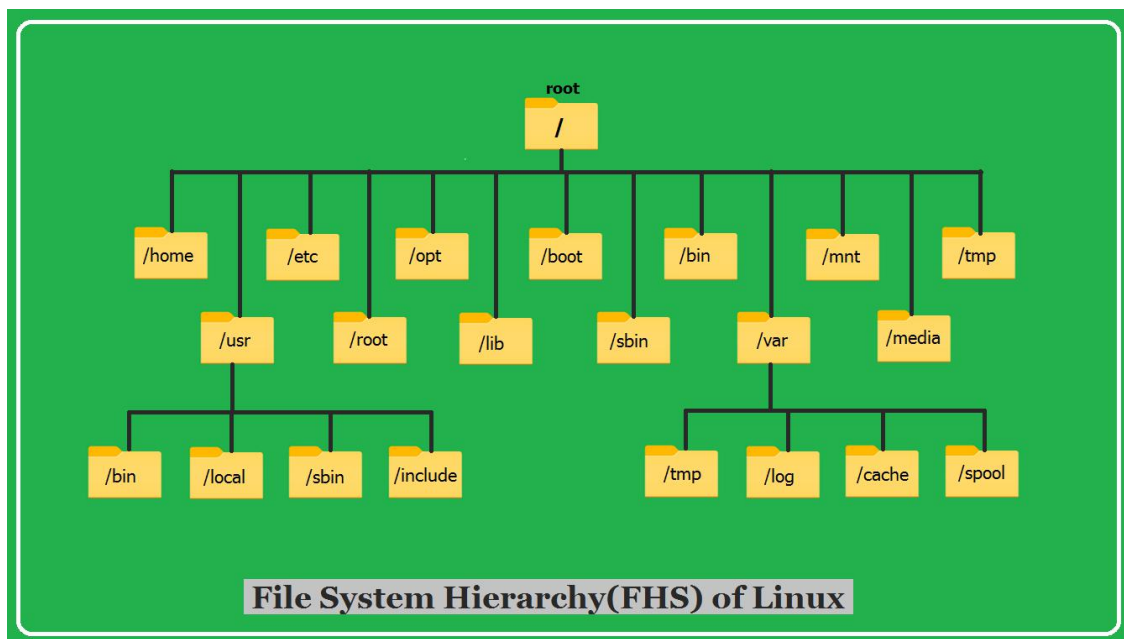
– The Root Directory
Everything on your Linux system is located under the / directory, known as the root directory.You can think of the / directory as being similar to the C:\ directory on Windows – but this isn't strictly true, as Linux doesn't have drive letters. While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.
/bin : Essential command binaries that need to be available in single user mode; for all users,
e.g., cat, ls, cp.

Contains binary executables

Common linux commands you need to use in single-user modes are located under this directory.

Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp

The /bin directory contains the essential user binaries (programs) that must be present when

the system is mounted in single-user mode. Applications such as Firefox are stored in /usr/bin,while important system programs and utilities such as the bash shell are located in /bin. The /usr directory may be stored on another partition – placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted. The /sbin directory is similar – it contains essential system administration binaries.

/boot – Static Boot File: The /boot directory contains the files needed to boot the system – for example, the GRUB boot loader's files and your Linux kernels are stored here. The boot

loader's configuration files aren't located here, though – they're in /etc with the other configuration files.

/cdrom – Historical Mount Point for CD-ROMs

The /cdrom directory isn't part of the FHS standard, but you'll still find it on Ubuntu and other operating systems. It's a temporary location for CD-ROMs inserted in the system. However, the standard location for temporary media is inside the /media directory.

/dev – Device Files

Linux exposes devices as files, and the /dev directory contains a number of special files that

represent devices. These are not actual files as we know them, but they appear as files – for

example, /dev/sda represents the first SATA drive in the system. If you wanted to partition it,

you could start a partition editor and tell it to edit /dev/sda.

This directory also contains pseudo-devices, which are virtual devices that don't actually

correspond to hardware. For example, /dev/random produces random numbers. /dev/null is a

special device that produces no output and automatically discards all input – when you pipe

the output of a command to /dev/null, you discard it.

/etc – Configuration Files

The /etc directory contains configuration files, which can generally be edited by hand in a text editor. Note that the /etc/ directory contains system-wide configuration files – user-specific configuration files are located in each user's home directory.

/home – Home Folders

The /home directory contains a home folder for each user. For example, if your user name is

bob, you have a home folder located at /home/bob. This home folder contains the user's data

files and user-specific configuration files. Each user only has written access to their own home

folder and must obtain elevated permissions (become the root user) to modify other files on the system.

/lib – Essential Shared Libraries

The /lib directory contains libraries needed by the essential binaries in the /bin and /sbin folder.Libraries needed by the binaries in the /usr/bin folder are located in /usr/lib.

/lost+found – Recovered Files

Each Linux file system has a lost+found directory. If the file system crashes, a file system

check will be performed at next boot. Any corrupted files found will be placed in the lost+found directory, so you can attempt to recover as much data as possible.

/media – Removable Media

The /media directory contains subdirectories where removable media devices inserted into the computer are mounted. For example, when you insert a CD into your Linux system, a directory will automatically be created inside the /media directory. You can access the contents of the CD inside this directory.

/mnt – Temporary Mount Points

Historically speaking, the /mnt directory is where system administrators mounted temporary

file systems while using them. For example, if you're mounting a Windows partition to perform

some file recovery operations, you might mount it at /mnt/windows. However, you can mount

other file systems anywhere on the system.

/opt – Optional Packages

The /opt directory contains subdirectories for optional software packages. It's commonly used

by proprietary software that doesn't obey the standard file system hierarchy – for example, a

proprietary program might dump its files in /opt/application when you install it.

/proc – Kernel & Process Files

The /proc directory similar to the /dev directory because it doesn't contain standard files. It

contains special files that represent system and process information.

/root – Root Home Directory

The /root directory is the home directory of the root user. Instead of being located at /home/root,

it's located at /root. This is distinct from /, which is the system root directory.

/run – Application State Files

The /run directory is fairly new, and gives applications a standard place to store transient files

they require like sockets and process IDs. These files can't be stored in /tmp because files in

/tmp may be deleted.

/sbin – System Administration Binaries

The /sbin directory is similar to the /bin directory. It contains essential binaries that are generally intended to be run by the root user for system administration

/selinux – SELinux Virtual File System

If your Linux distribution uses SELinux for security (Fedora and Red Hat, for example), the

/selinux directory contains special files used by SELinux. It's similar to /proc. Ubuntu doesn't

use SELinux, so the presence of this folder on Ubuntu appears to be a bug.

/srv – Service Data

The /srv directory contains "data for services provided by the system." If you were using the

Apache HTTP server to serve a website, you'd likely store your website's files in a directory

inside the /srv directory.

/tmp – Temporary Files

Applications store temporary files in the /tmp directory. These files are generally deleted

whenever your system is restarted and may be deleted at any time by utilities such as tmpwatch**.**

/usr – User Binaries & Read-Only Data

The /usr directory contains applications and files used by users, as opposed to applications and files used by the system. For example, non-essential applications are located inside the /usr/bin directory instead of the /bin directory and non-essential system administration binaries are located in the /usr/sbin directory instead of the /sbin directory. Libraries for each are located inside the /usr/lib directory. The /usr directory also contains other directories – for example,architecture-independent files like graphics are located in /usr/share.The /usr/local directory is where locally compiled

applications install to by default – this prevents them from mucking up the rest of the system.

/var – Variable Data Files
The /var directory is the writable counterpart to the /usr directory, which must be read-only in
normal operation. Log files and everything else that would normally be written to /usr during
normal operation are written to the /var directory. For example, you'll find log files in /var/log.

6. Installation and configuration of LAMP stack. Deploy an open source application such as phpMyAdmin and WordPress. Procedure
Install Apache2

Update your system:

sudo apt update

```
mca@S3:~$  sudo apt update
Hit:1 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Err:3 http://ppa.launchpad.net/jonathonf/python-3.6/ubuntu bionic InRelease
  403  Forbidden [IP: 185.125.190.52 80]
Ign:4 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 InRelease
Hit:5 http://ppa.launchpad.net/webupd8team/java/ubuntu bionic InRelease
Get:6 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 Release [2,495 B]
Get:7 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 Release.gpg [801 B]
Err:7 https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.6 Release.gpg
```

Install Apache using apt:
sudo apt install apache2

```
mca@S3:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.29-1ubuntu4).
The following packages were automatically installed and are no longer required:
  debhelper dh-autoreconf dh-strip-nondeterminism libarchive-cpio-perl libfile-stripnondeterminism-perl libmail-sendmail-perl libpcre16-3
  libpcre3-dev libpcre32-3 libpcrecpp0v5 libssl-dev libssl-doc libsys-hostname-long-perl php-common php-pear php-xml php7.2-cli
  php7.2-common php7.2-json php7.2-opcache php7.2-readline php7.2-xml pkg-php-tools po-debconf shtool
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
```

Confirm that Apache is now running with the following command:
sudo systemctl status apache2
If it is not working !
sudo systemctl stop apache2 # to stop if running
sudo systemctl start apache2 # to start if not running
Once installed, test by accessing your server's IP in your browser:
http://127.0.0.1/
http://localhost/
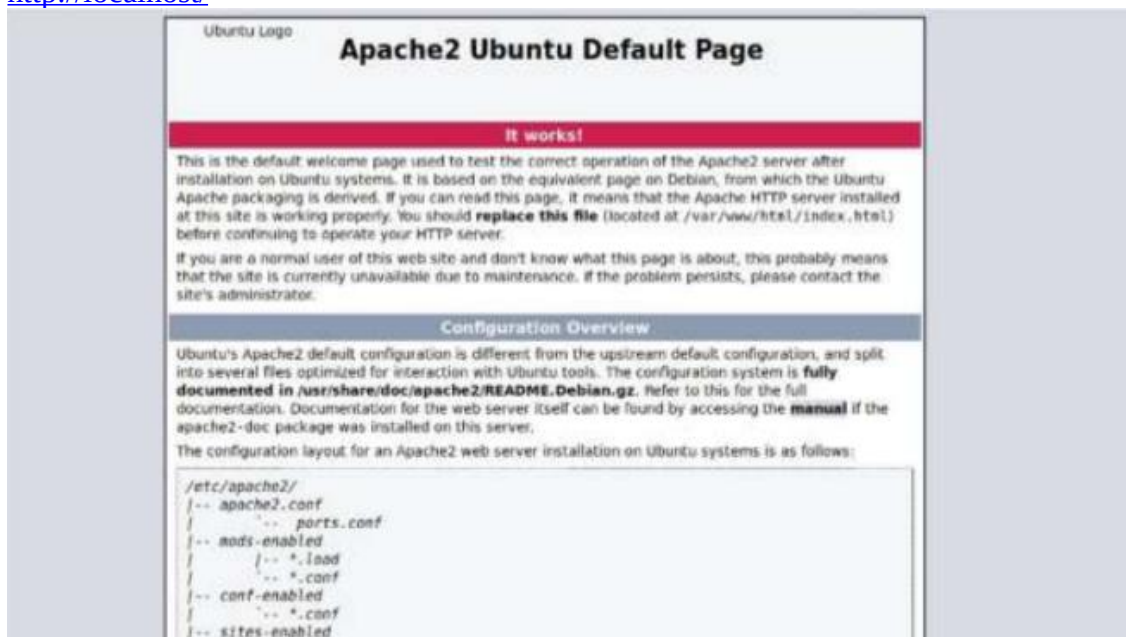
Install mariadb

sudo apt install mariadb-server mariadb-client
sudo systemctl status mysql # to check status
sudo systemctl start mysq # if not running
sudo mysql_secure_installation # Secure your newly installed MariaDB

Install PHP and commonly used modules
    sudo apt install php libapache2-mod-php php-opcache php-cli php-gd
php- curl php-mysql
    sudo systemctl restart apache2
    Test PHP Processing on Web Server
sudo nano /var/www/html/phpinfo.php
    Inside the file, type in the valid PHP code:
<?php
phpinfo ();
?>
    **Press CTRL + X to save and close the file. Press y and ENTER to
confirm Open a browser and type in your IP address/phpinfo.php**
http://127.0.0.1/phpinfo.php
Install phpmyadmin
sudo apt install phpmyadmin php-mbstring php-zip php-gd php-json php-
curl
sudo systemctl restart apache2
    Open a browser: http://localhost/phpmyadmin
username:root
password: yourpassword If php my admin page not found:
nano /etc/apache2/apache2.conf
    Add this line to last of the file.
**Press CTRL + X to save and close the file. Press y and ENTER to
confirm**
Include /etc/phpmyadmin/apache.conf
    restart apache2 - now try: http://localhost/phpmyadmin
sudo systemctl restart apache2
    If any problem for login run the following command
sudo mysql
ALTER USER root@localhost IDENTIFIED BY "yourpassword";
Install WordPress with LAMP on Ubuntu 18.04
Step 1 – Download WordPress
Download the latest version of the WordPress package and extract it by
issuing the commands below on the terminal:
    wget -c http://wordpress.org/latest.tar.gz

```
mca@S3:~$ wget -c http://wordpress.org/latest.tar.gz
--2022-06-13 15:24:19--  http://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://wordpress.org/latest.tar.gz [following]
--2022-06-13 15:24:20--  https://wordpress.org/latest.tar.gz
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 21166276 (20M) [application/octet-stream]
Saving to: 'latest.tar.gz'

latest.tar.gz                   100%[=====================================

2022-06-13 15:24:24 (5.98 MB/s) - 'latest.tar.gz' saved [21166276/21166276]
```

    tar -xzvf latest.tar.gz

```
mca@S3:~$ tar -xzvf latest.tar.gz
wordpress/
wordpress/xmlrpc.php
wordpress/wp-blog-header.php
wordpress/readme.html
wordpress/wp-signup.php
wordpress/index.php
wordpress/wp-cron.php
wordpress/wp-config-sample.php
wordpress/wp-login.php
wordpress/wp-settings.php
wordpress/license.txt
wordpress/wp-content/
wordpress/wp-content/themes/
wordpress/wp-content/themes/twentytwentyone/
wordpress/wp-content/themes/twentytwentyone/footer.php
wordpress/wp-content/themes/twentytwentyone/template-parts/
```

Then move the WordPress files from the extracted folder to the Apache
default root directory, /var/www/html/:

    sudo mv wordpress/* /var/www/html/

Next, set the correct permissions on the website directory, that is give
ownership of the WordPress files to the webserver as follows:

    sudo chown -R www-data:www-data /var/www/html/
    sudo chmod -R 755 /var/www/html/

Step 2 – Creating a MySQL Database and User for WordPress

The first step you'll take is a preparatory one. Even though MySQL is already installed, you still need to create a database to manage and store the user information for WordPress to use. To get started, log into the MySQL root(administrative) account by issuing the following command:

    sudo mysql
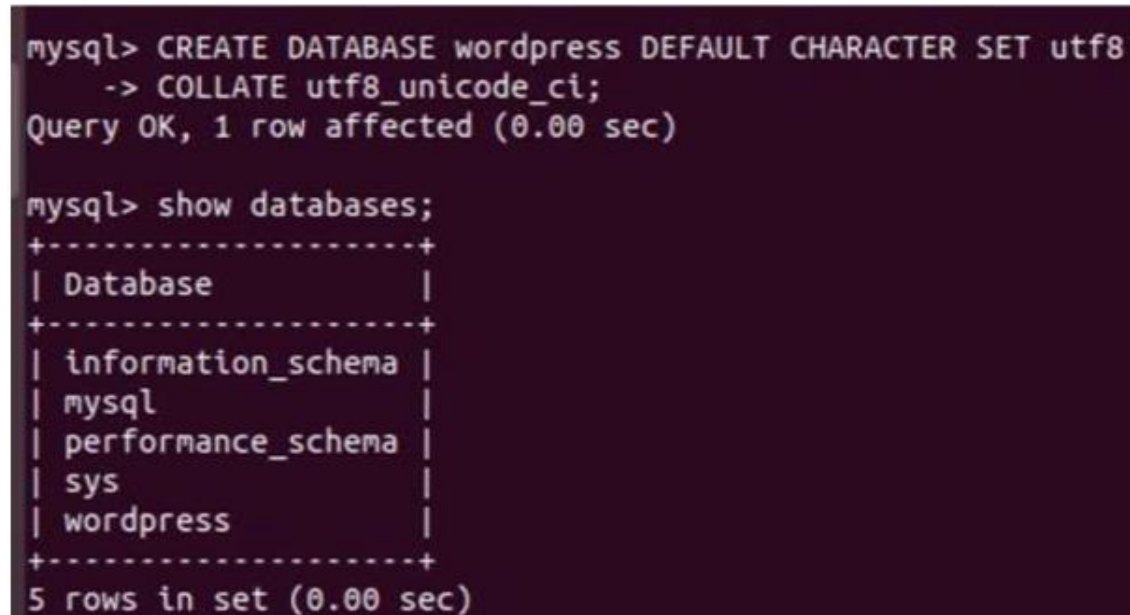
You will be prompted for the password you set for the MySQL root accountwhen you installed the software. However, if you have password authenticationenabled for your

root user, you can run the following command and enter your password information when prompted:

mysql -u root –p

From there, you'll create a new database that WordPress will control. You can call this whatever you would like, but we will be using wordpress in this guideas an example. Create the database for WordPress by writing the following:

CREATE DATABASE wordpress DEFAULT CHARACTER
SET utf8 COLLATE utf8_unicode_ci;

```
mysql> CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8
    -> COLLATE utf8_unicode_ci;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wordpress          |
+--------------------+
5 rows in set (0.00 sec)
```

Next, you're going to create a separate MySQL user account that you'll useexclusively to operate on the new database. Creating one-function databases andaccounts is a good idea from a management and security standpoint. We willuse the name wordpressuser as an example in this guide. Feel free to change thisif you'd like.You can create this account, set a password for it, and then grant it access to thedatabase you created all by running the following command. Remember tochoose a strong password here for your database user:

GRANT ALL ON wordpress.* TO
'wordpressuser'@'localhost'IDENTIFIED BY 'password';
After creating this user, flush the privileges to ensure that the current
instance ofMySQL knows about the recent changes you've made:

FLUSH PRIVILEGES;
Exit out of MySQL:

EXIT
You now have a database and user account in MySQL, each made
specifically for WordPress. Go the /var/www/html/ directory and rename
existing wp-config-sample.php to wpconfig.php. Also, make sure to
remove the default Apache index page.

cd /var/www/html/
sudo mv wp-config-sample.php wp-config.php
sudo rm -rf index.html

```
mca@S3:~$ cd /var/www/html/
mca@S3:/var/www/html$ sudo mv wp-config-sample.php wp-config.php
mca@S3:/var/www/html$ sudo rm -rf index.html
```

Then update it with your database information under the MySQL settings section (refer to the highlighted boxes in the image below):This setting can be added after the database connection settings, or anywhere else in the file: Save and close the file when you are finished.Restart the web server and mysql service using the commands below:

sudo systemctl restart apache2.service

sudo systemctl restart mysql.service

```
mca@S46:/var/www/html$ sudo systemctl restart apache2.service
mca@S46:/var/www/html$ sudo systemctl restart mysql.service
```

Step 3 – Completing the Installation Through the Web Interface

Now that the server configuration is complete, you can complete the installation through the web interface. In your web browser, navigate to your server's domain name or public IP address:

https://server_domain_or_IP

Select the language you would like to use: Next you will be directed to the main setup page. Select a name for your WordPress site and choose a username (it is recommended not to choose something like "admin" for security purposes). A strong password is generated automatically. Save this password or select an alternative strong password.Enter your email address and select whether you want to discourage search engines from indexing your site:Once you log in, you will be taken to the WordPress administration dashboard: From there, you can begin using and customizing your WordPress site.
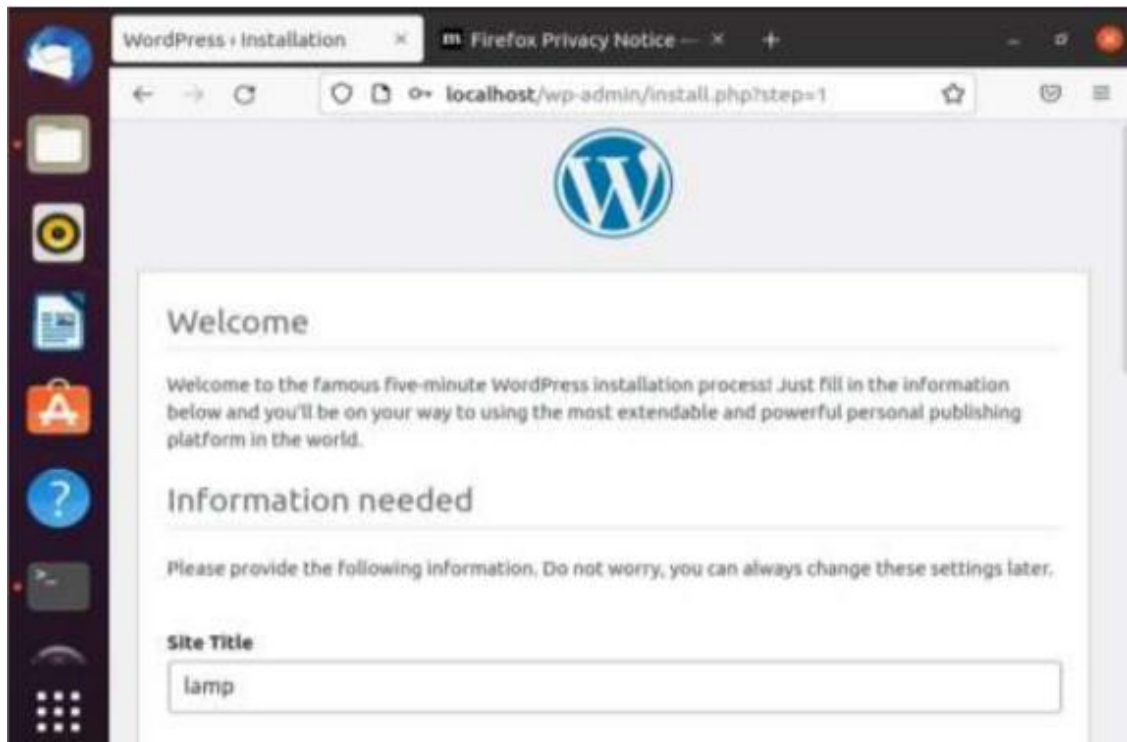
Once you log in, you will be taken to the WordPress administration dashboard: From there, you can begin using and customizing your WordPress site.
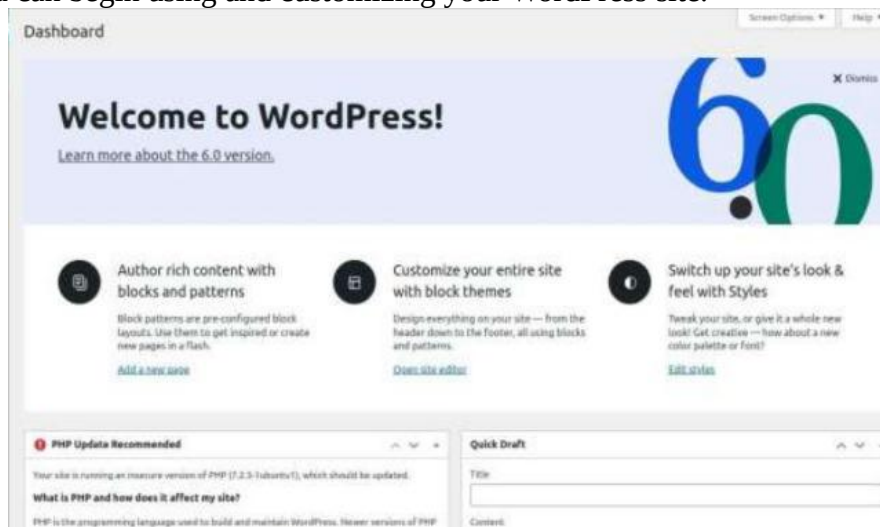
7. Build and install software from source code, familiarity with make and cmake utilities expected.

Procedure& Output Screenshot:
Install the cmake
Apt show cmake

```
mca@Z238-UL:       $ apt show cmake
Package: cmake
Version: 3.16.3-1ubuntu1.20.04.1
Priority: optional
Section: devel
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Debian CMake Team <pkg-cmake-team@lists.alioth.debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 19.2 MB
Depends: cmake-data (= 3.16.3-1ubuntu1.20.04.1), procps, libarchive13 (>= 3.3.3), libc6
oncpp1 (>= 1.7.4), librhash0 (>= 1.2.6), libstdc++6 (>= 9), libuv1 (>= 1.11.0), zlib1g (
Recommends: gcc, make
Suggests: cmake-doc, ninja-build
Homepage: https://cmake.org/
Download-Size: 3,668 kB
```

   $sudo apt install cmake g++ make: To install cmake , g++ and make using the apt command.

```
mca@Z238-UL:       $ sudo apt install cmake g++ make
[sudo] password for mca:
Reading package lists... Done
Building dependency tree
Reading state information... Done
g++ is already the newest version (4:9.3.0-1ubuntu2).
g++ set to manually installed.
make is already the newest version (4.2.1-1.2).
make set to manually installed.
The following packages were automatically installed and are no longer required:
  gcc-8-base:i386 intltool libaribb24-0 libctemplate3 libdrm-dev libdvdread7
  libevent-core-2.1-6 libglvnd-core-dev libgnome-keyring-common libicu-le-hb0
  libkate1 libmad0 libmysqlclient20 libmysqlcppconn7v5 libnfs11 libonig4
  libpciaccess-dev libplacebo7 libproxy-tools libresid-builder0c2a
  libsdl-image1.2 libsdl1.2debian libtinfo-dev libva-wayland2 libx265-146
  libxcb-dri2-0-dev libxcb-sync-dev linux-headers-5.4.0-167
  linux-headers-5.4.0-167-generic linux-image-5.4.0-167-generic
  linux-modules-5.4.0-167-generic linux-modules-extra-5.4.0-167-generic
  mesa-common-dev python-cffi-backend python-enchant python-ipaddress
  python-pexpect python-ptyprocess python-renderpm python-reportlab-accel
  python-webencodings vlc-data x11proto-dri2-dev x11proto-fixes-dev
  x11proto-gl-dev
```

Create directory

   Mkdir cmake: creating a different directory for our project usingthe mkdir and cd commands.

```
mca@Z238-UL:       $ mkdir myproject
```

   Cd cmake

```
mca@Z238-UL:       $ cd myproject
```

   gedit Helloworld.cpp
Now create a C++ source file named Hello_world.cpp and add the following **: gedit CmakeLists.txt** Create a CMakeLists.txt file(with this exact capitalization) which is required by CMake:

Create directory called

Mkdir build:
To run cmake we need to change into the build directory:
Cmake..

Cmake –build : To generate the executable simply by typing:run hello

```
mca@S3:~/Documents/CMake/myproject/build$ cmake --build .
Scanning dependencies of target hello
[ 50%] Building CXX object CMakeFiles/hello.dir/Hello_world.cpp.o
[100%] Linking CXX executable hello
[100%] Built target hello
```

./hello: Run the executable by typing:

```
mca@S3:~/Documents/CMake/myproject/build$ ./hello
Hello World!
```

8. Introduction to command line tools for networking IPv4 networking, network commands: ping route traceroute, nslookup, ip. Setting up static and dynamic IP addresses. Concept of Subnets, CIDR address schemes, Subnet masks, iptables, setting
up a firewall for LAN, Application layer (L7) proxies.

Procedure:

a. **ifconfig**:This commands in windows allows you to see a summarized information of your
network such as ip address, subnet mask , server address etc.
Output

```
sjcet@Z238-UL:~/kishor/sem2/CN$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.54.197  netmask 255.255.240.0  broadcast 172.16.63.255
        inet6 fe80::a4fb:b17f:9247:b333  prefixlen 64  scopeid 0x20<link>
        ether 3c:52:82:6e:b2:a7  txqueuelen 1000  (Ethernet)
        RX packets 366691  bytes 298946338 (298.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 188619  bytes 49238714 (49.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xd1000000-d1020000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 5842  bytes 546463 (546.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5842  bytes 546463 (546.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

b. nslookup : To show the server to which the system is connected by default. If we want to
find the ip address of a particular domain name, we can also use nslookup

```
sjcet@Z238-UL:~/kishor/sem2/CN$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.193.142
Name:   google.com
Address: 2404:6800:4007:81f::200e
```

c. ping : The command used to check the availability of a host. The response shows the URL
you are pinging, the ip address associated with the URL and the size of packets being sent on
the first line . The next four lines shows the replies from each individual packets
including the time(in milliseconds) for the response and the time to live(TLL) of the packet, that is the
amount of time that must pass before the packet discarded.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ ping -c 4 google.com
PING google.com (142.250.196.46) 56(84) bytes of data.
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=1 ttl=59 time
=70.9 ms
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=2 ttl=59 time
=71.8 ms
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=3 ttl=59 time
=73.5 ms
64 bytes from maa03s45-in-f14.1e100.net (142.250.196.46): icmp_seq=4 ttl=59 time
=71.3 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 70.942/71.885/73.494/0.972 ms
sjcet@Z238-UL:~/kishor/sem2/CN$ █
```

d. traceroute : traceroute is a command-line utility in Linux and other Unix-like operating

systems that allows you to track the path that packets take from your computer to a destination host on a network. It's used for troubleshooting network connectivity issues and identifying network delays

```
sjcet@Z238-UL:~/kishor/sem2/CN$ traceroute google.com
traceroute to google.com (142.250.195.46), 30 hops max, 60 byte packets
 1  _gateway (172.16.48.2)  0.250 ms  0.197 ms  0.153 ms
 2  172.24.71.66 (172.24.71.66)  1.801 ms  1.868 ms  1.827 ms
 3  * * *
 4  * * *
 5  72.14.218.250 (72.14.218.250)  39.904 ms  40.840 ms *
 6  * * *
 7  142.251.55.74 (142.251.55.74)  40.436 ms * *
 8  142.251.55.67 (142.251.55.67)  33.950 ms 74.125.242.138 (74.125.242.138)  18
.648 ms 74.125.242.147 (74.125.242.147)  32.593 ms
 9  108.170.253.113 (108.170.253.113)  39.160 ms maa03s37-in-f14.1e100.net (142.
250.195.46)  20.368 ms 108.170.253.113 (108.170.253.113)  38.347 ms
sicet@Z238-UL:~/kishor/sem2/CN$ █
```

e. netstat : netstat is a command-line utility in Linux and other Unix-like operating systems

that provides information about network connections, routing tables, interface statistics, masquerade connections, and more. It's used for monitoring network-related information and

diagnosing network issues.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address          State
tcp        0      0 localhost:domain       0.0.0.0:*                LISTEN
tcp        0      0 localhost:ipp          0.0.0.0:*                LISTEN
tcp        0      0 localhost:33066        0.0.0.0:*                LISTEN
tcp        0      0 localhost:mysql        0.0.0.0:*                LISTEN
tcp        0      0 Z238-UL:50362          sd-in-f188.1e100.n:5228  ESTABLISHED
tcp        0      0 Z238-UL:41866          maa03s46-in-f10.1:https  ESTABLISHED
tcp        0      0 Z238-UL:56220          maa03s42-in-f3.1e:https  ESTABLISHED
tcp        0      0 Z238-UL:45476          maa05s18-in-f10.1:https  ESTABLISHED
tcp        0      0 Z238-UL:38946          162.247.243.29:https     ESTABLISHED
tcp        0      0 Z238-UL:39036          maa03s34-in-f10.1:https  ESTABLISHED
tcp        0      0 Z238-UL:35500          maa05s09-in-f3.1e:https  ESTABLISHED
tcp        0      0 Z238-UL:52014          104.18.2.161:https       ESTABLISHED
tcp        0      0 Z238-UL:33820          a184-51-195-169.d:https  ESTABLISHED
tcp        0  29587 Z238-UL:44876          maa05s17-in-f14.1:https  ESTABLISHED
tcp        0      0 Z238-UL:42126          maa05s28-in-f14.1:https  ESTABLISHED
tcp        0      0 Z238-UL:35788          maa05s09-in-f14.1:https  ESTABLISHED
tcp        0      0 Z238-UL:56332          84.170.224.35.bc.g:http  TIME_WAIT
tcp        0      1 Z238-UL:37450          maa05s12-in-f4.1e:https  LAST_ACK
tcp        0      0 Z238-UL:47766          maa03s26-in-f3.1e:https  ESTABLISHED
tcp        0      0 Z238-UL:48350          maa03s46-in-f14.1:https  TIME_WAIT
tcp        0      0 Z238-UL:46120          maa03s42-in-f3.1e:https  ESTABLISHED
tcp        0      0 Z238-UL:45488          maa05s18-in-f10.1:https  ESTABLISHED
tcp        0      0 Z238-UL:58958          li781-4.members.1:https  ESTABLISHED
tcp        0      1 Z238-UL:57628          maa05s20-in-f4.1e:https  LAST_ACK
tcp        0      0 Z238-UL:40184          maa05s17-in-f13.1:https  TIME_WAIT
tcp        0      0 Z238-UL:53738          172.16.208.2:8090        ESTABLISHED
tcp        0      0 Z238-UL:52980          li695-222.members:https  ESTABLISHED
tcp        0      0 Z238-UL:49846          maa05s25-in-f3.1e:https  ESTABLISHED
tcp        0      0 Z238-UL:60996          ec2-35-174-127-31:https  ESTABLISHED
tcp        0      0 Z238-UL:44324          maa05s19-in-f14.1:https  ESTABLISHED
tcp6       0      0 ip6-localhost:ipp      [::]:*                   LISTEN
udp        0      0 224.0.0.251:mdns       0.0.0.0:*
udp        0      0 224.0.0.251:mdns       0.0.0.0:*
udp        0      0 0.0.0.0:mdns           0.0.0.0:*
udp        0      0 0.0.0.0:42940          0.0.0.0:*
udp        0      0 localhost:domain       0.0.0.0:*
udp        0      0 Z238-UL:bootpc         _gateway:bootps          ESTABLISHED
udp        0      0 0.0.0.0:631            0.0.0.0:*
udp        0      0 localhost:domain       0.0.0.0:*
udp        0      0 Z238-UL:bootpc         _gateway:bootps          ESTABLISHED
udp        0      0 0.0.0.0:631            0.0.0.0:*
udp6       0      0 [::]:mdns              [::]:*
```

f. hostname : The hostname command is a command-line utility in Linux and other Unix-like
operating systems that allows you to view or set the hostname of the system. The hostname is
the unique name assigned to a computer within a network.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ hostname
Z238-UL
```

g. arp : The arp command is a command-line utility in Linux and other Unix-like operating
systems that allows you to view and manipulate the Address Resolution Protocol (ARP)
cache, which is used to map IP addresses to MAC addresses on a local network. ARP is
essential for communication between devices within the same subnet.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ arp -a
_gateway (172.16.48.2) at 7c:5a:1c:cf:50:b9 [ether] on eno1
```

h. uname : The uname command is a command-line utility in Linux and other Unix-like
operating systems that provides information about the system's kernel and operating system.
It's used to retrieve information about the system's architecture, release version, and other
details.

```
sjcet@Z238-UL:~/kishor/sem2/CN$ uname -a
Linux Z238-UL 5.4.0-147-generic #164-Ubuntu SMP Tue Mar 21 14:23:17 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

9. Analyzing network packet stream using tcpdump and wireshark. Perform basic network service tests using nc.

Procedure:
a. How to Install tcpdump in Linux
Many Linux distributions already shipped with the tcpdump tool, if in case you don't have it on a system, you can install it using the command.
    $ sudo apt-get install tcpdump [On Debian, Ubuntu and Mint]

b. Display Available Interfaces

To list the number of available interfaces on the system, run the following command with -D option.

c. Capture Packets from Specific Interface
The command screen will scroll up until you interrupt and when we execute the tcpdump command it will captures from all the interfaces, however with -i switch only capture from the desired interface.

d. Capture Only N Number of Packets
When you run the tcpdump command it will capture all the packets for the specified interface, until you hit the cancel button. But using -c option, you can capture a specified number of packets.

tcpdump –c 5 –i enp3s0

e. Display Captured Packets in HEX and ASCII
The following command with option -XX capture the data of each packet, including its link level header in HEX and ASCII format

f. Capture and Save Packets in a File
As we said, that tcpdump has a feature to capture and save the file in a .pcap format, to do this just execute the command with -w option.

g. Capture Packet from Specific Port
Let's say you want to capture packets for specific port 80, execute the below command by specifying port number 80

h. Read Captured Packets File
To read and analyze captured packet 0001.pcap file use the command with -r
option
wire shark
Installing Wireshark on Ubuntu 20.04
The Wireshark utility is available on all major desktop platforms, i.e., Linux, Microsoft Windows, FreeBSD, MacOS, Solaris, and many more. Follow the steps below to install Wireshark on Ubuntu 20.04.
Step 1 : Update APT
First, as always, update and upgrade your APT through the following command.
Syntax:
$ sudo apt update

Step 2: Download and Install Wireshark

Now that Wireshark's latest version has been added to the APT, you can download and install it with the following command.
syntax
$ sudo apt install wireshark

Step 3: Enable Root Privileges
When Wireshark installs on your system, you will be prompted by the following window. As Wireshark requires superuser/root privileges to operate, this option asks to enable or disable permissions for all every user on the system. Press the "Yes" button to allow other users, or press the "No" button to restrict other users from using Wireshark.

Step 4:
You must add a username to the Wireshark group so that this user can use Wireshark. To do this, execute the following command, adding your required username after "wireshark" in the command.
Syntax:
$ sudo adduser $user wireshark

Step 5: Launch Wireshark
In the terminal window, type the following command to start the Wireshark application.
 Syntax:
$ wireshark
You can also open Wireshark through the Graphical User Interface (GUI) by opening the activities on the Ubuntu desktop, and in the search bar, type "Wireshark," and click on the application result.

10.a) Introduction to Hypervisors and VMs, Xen or KVM .

Procedure:
For the Ubuntu system, all packages required to run KVM are available on official upstream
repositories.
Install them using the commands:
    sudo apt update
    apt-get install qemu qemu-kvm libvirt-bin bridge-utils virt-manager virtviewer–
y

Create Virtual Machine • You can create virtual machine using virt-manager utility. Run the
following command to start the virt-manager:
    sudo virt-manager
    virsh help
    virsh help
    virsh help list
    Sudo virsh nodeinfo
    Virsh start
    vm virsh start
    virsh start testvm1
Step 1: Update the repositories
Step 2: Install essential KVM packages
Install virt-manager, a tool for creating and managing VMs

Step 3: Start virt-manager with
Step 4: In the first window, click the computer icon in the upper-left corner,In the dialogue box
that opens, select the option to install the VM using an ISO image. Then click Forward.

Step 5: Choose ISO, click Forward
Step 6: Enter the amount of RAM and the number of CPUs you wish to allocate to the VM and
            proceed to the next step.
Step 7: Allocate hard disk space to the VM. Click Forward to go to the last step.

Step 8: Specify the name for your VM and click Finish to complete the setup.

Step 9: Select language
Step 10: The VM starts automatically, prompting you to start installing the OS that's on the ISO file.
Step 11: Check the state of KVM

b) Introduction to Containers: Docker installation and deployment

Procedure:
Steps for Installing Docker:
Step 1 : Open the terminal on Ubuntu.
Step 2 : Remove any Docker files that are running in the system, using the following command
    Command: $ sudo apt-get remove docker docker-engine docker.io
After entering the above command, you will need to enter the password of the root and press enter.
Step 3 : Check if the system is up-to-date using the following command:
    Command: $ sudo apt-get update
Step 4 : Install Docker using the following command:
    Command: $ sudo apt install docker.io
You'll then get a prompt asking you to choose between y/n – choose 'y'
Step 5 : Install all the dependency packages using the following command:
    Command: $ sudo snap install docker
Step 6 : Before testing Docker, check the version installed using the following command:
    Command: $ docker –version
Step 7 : Pull an image from the Docker hub using the following command:
    Command: $ sudo docker run hello-world
Here, hello-world is the docker image present on the Docker hub.
Step 8 : Check if the docker image has been pulled and is present in your system using the
following command:
    Command: $ sudo docker images

Step 9 : To display all the containers pulled, use the following command:
    Command : $ sudo docker ps -a
Step 10 : To check for containers in a running state, use the following command:
    Command: $ sudo docker ps

11. Installing and configuring modern frameworks like Laravel typically involves setting up a web server, PHP, a database, and the framework itself. Below is a general guide to installing and configuring Laravel on a Linux system. Please note that specific steps may vary based on your distribution and environment.

Step 1: Prerequisites

Install Required Software: Make sure you have a web server (e.g., Apache or Nginx), PHP, Composer (dependency manager), and a database server (e.g., MySQL) installed on your system.

Install Composer: Download and install Composer by following the instructions on the official Composer website.

Step 2: Install Laravel

1. Create a New Laravel Project: Open a terminal and navigate to the directory where you want to create your Laravel project. Run the following command:

composer create-project --prefer-dist laravel/laravel myproject

This will create a new Laravel project named "myproject."

Step 3: Configure the Web Server

1. Apache:

Create a new virtual host configuration for your Laravel project in your Apache configuration.

Set the DocumentRoot to the public directory of your Laravel project.

Enable the necessary Apache modules (e.g., rewrite) and restart Apache.

2. Nginx:

Create a new server block configuration for your Laravel project in your Nginx configuration.

Set the root directive to the public directory of your Laravel project.

Configure the necessary location directives and restart Nginx.

Step 4: Configure Laravel

1. Environment Configuration:

Rename the .env.example file in your Laravel project root to .env.

Set database connection details, application key, and other settings in the .env File.

2. Generate Application Key: Run the following command in your Laravel project directory:

php artisan key:generate

3. Run Migrations: If your .env file is configured with database details, run migrations to create necessary database tables:

php artisan migrate

Step 5: Testing the Setup

1. Access the Application: Open a web browser and visit the URL you configured for your Laravel project. You should see the Laravel welcome page.

2. Create Routes and Views: Begin building your application by defining routes and creating views in the resources/views directory.