18CSC205J - Operating Systems

LAB MANUAL

Bachelor of Technology

Semester IV

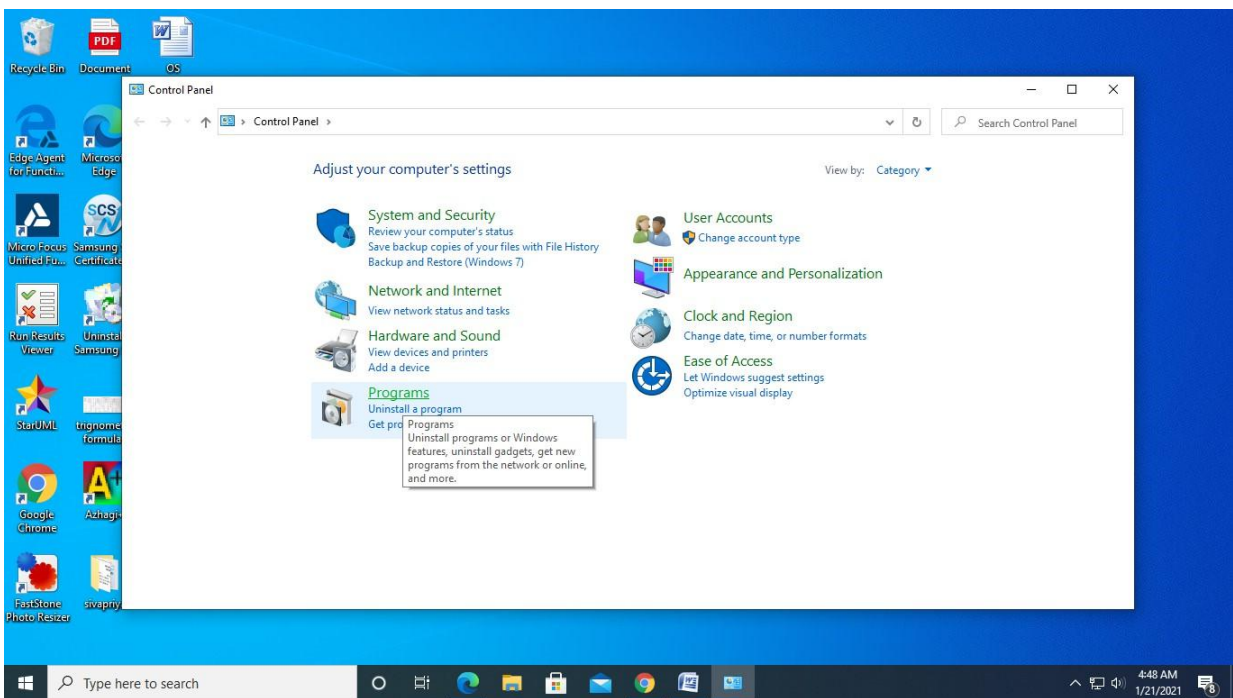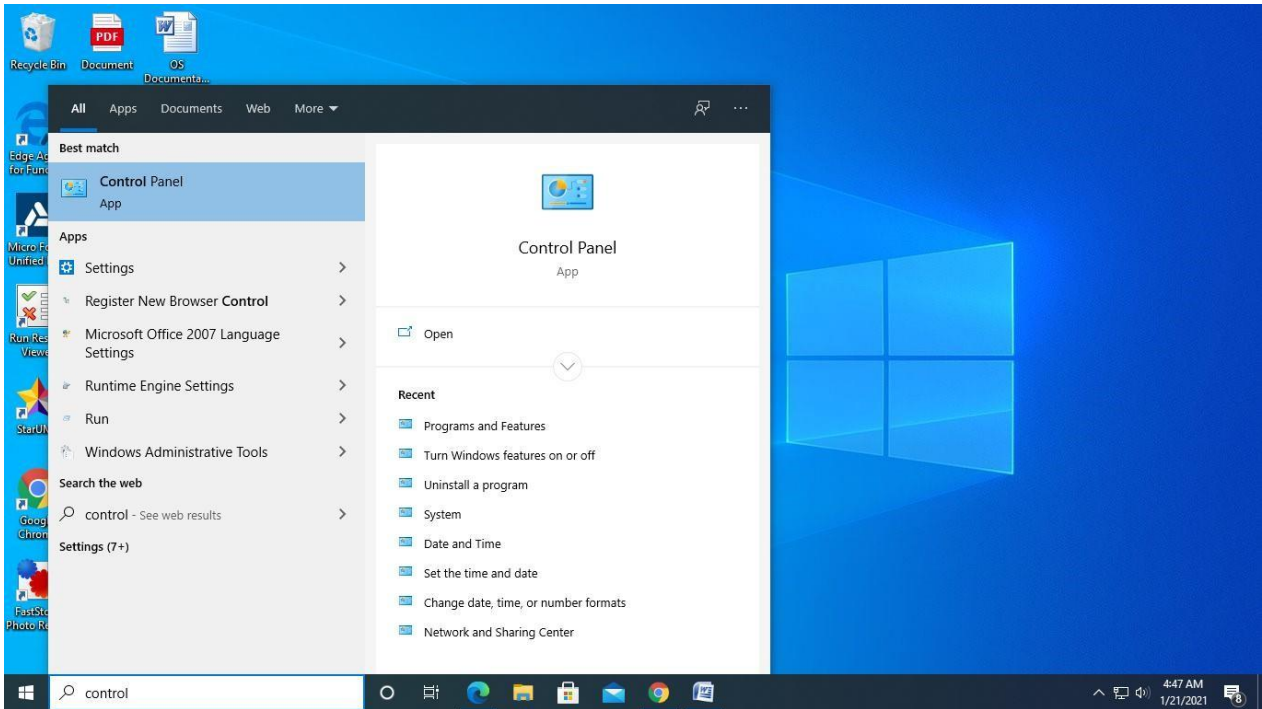## Academic Year: 2021-2022 EVEN SEMESTER

FACULTY OF ENGINEERING AND TECHNOLOGY

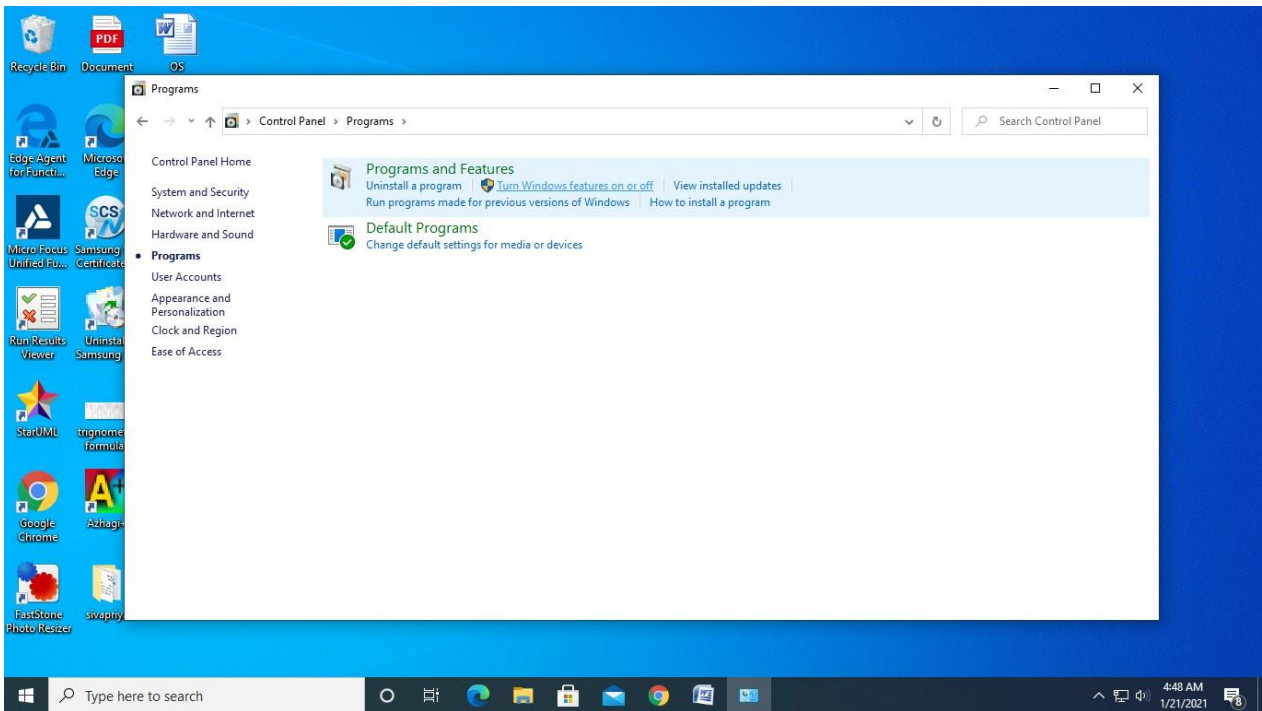**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

RAMAPURAM CAMPUS,
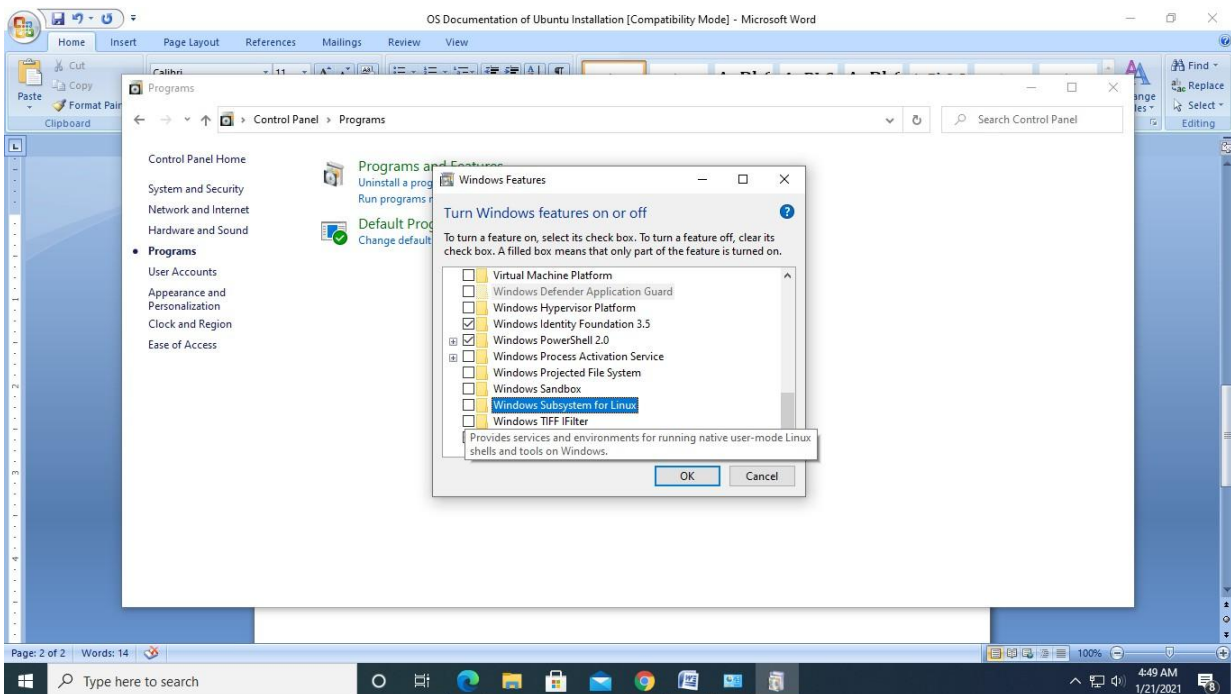
CHENNAI- 600 089

Ubuntu installation guidelines in windows

Installing Ubuntu in windows 10- 64 bit
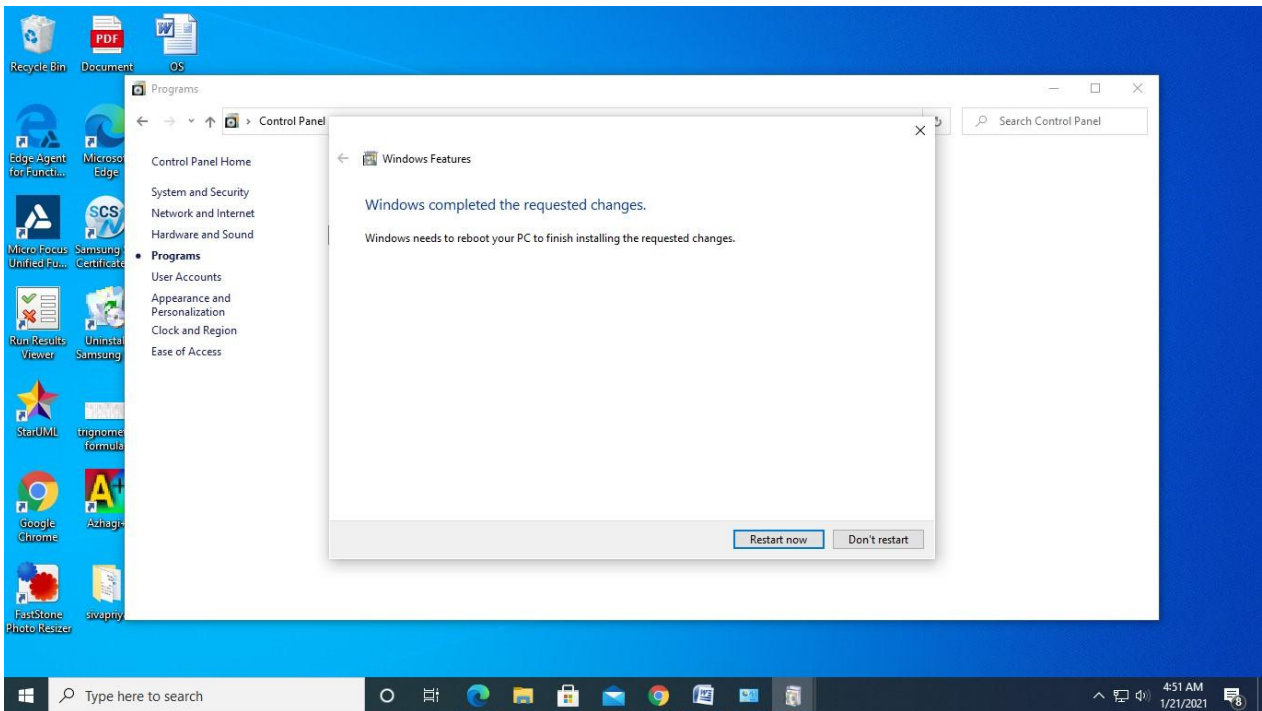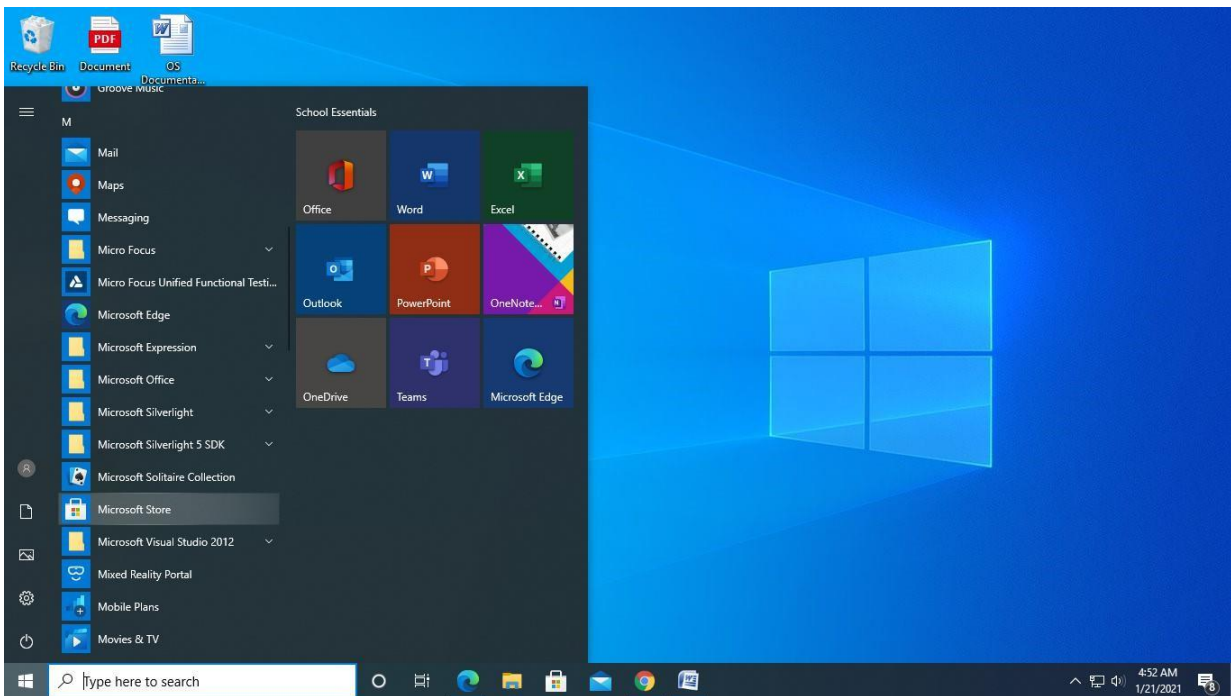
Select Turn Windows features On or Off

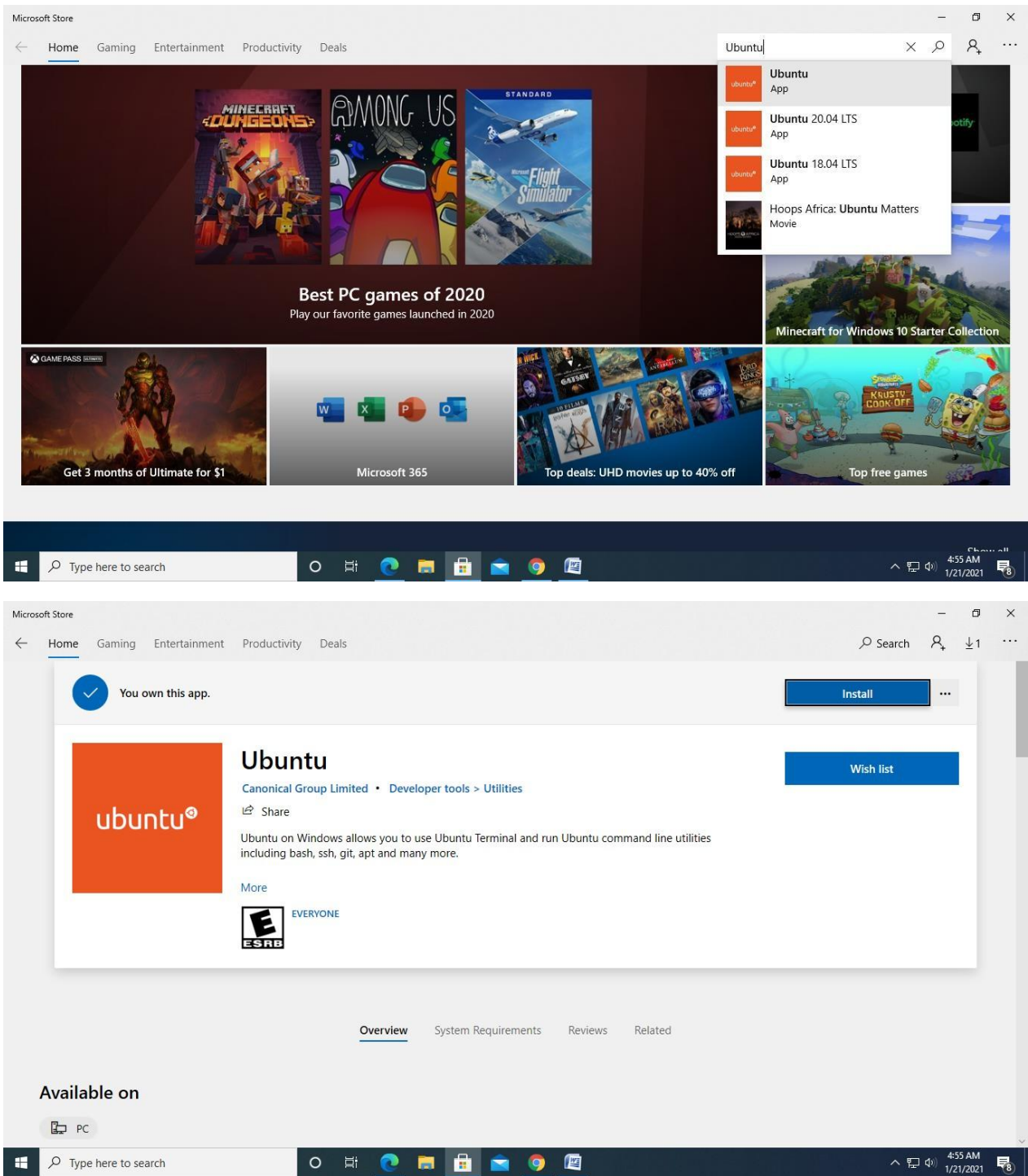
Select Windows subsystem for linux then press Ok

Now restart the PC to apply the changes

Choose Microsoft Store and search for Ubuntu and Install it

Click on Install
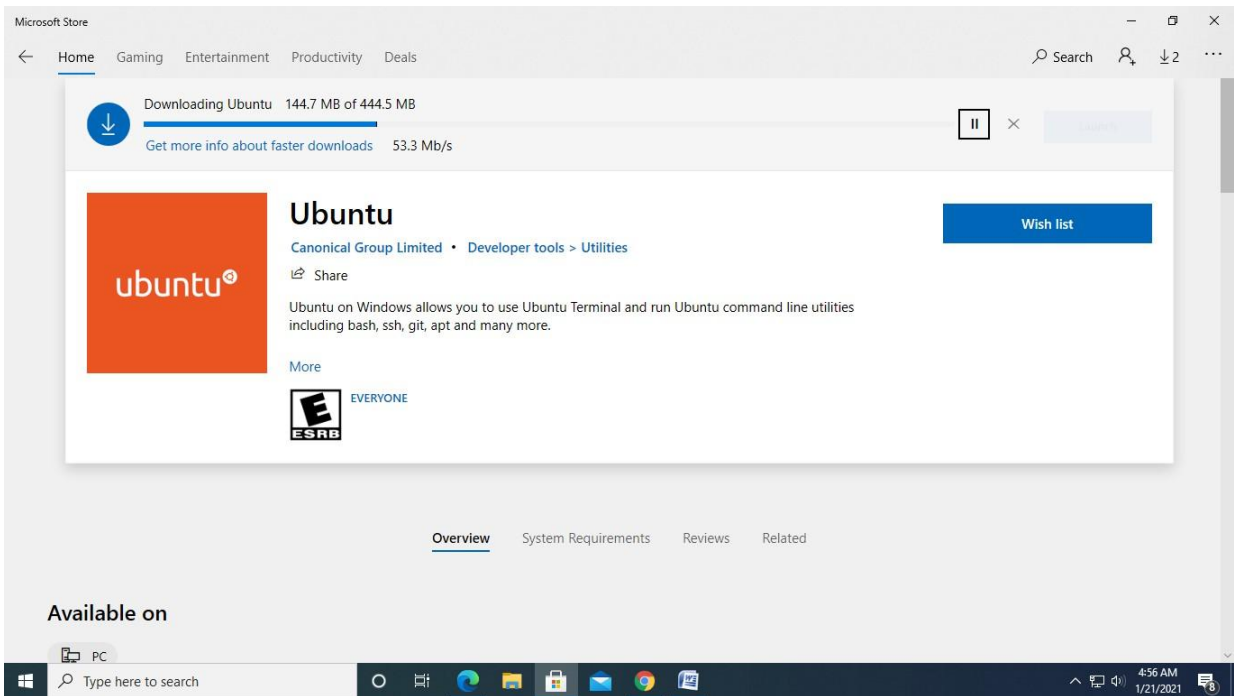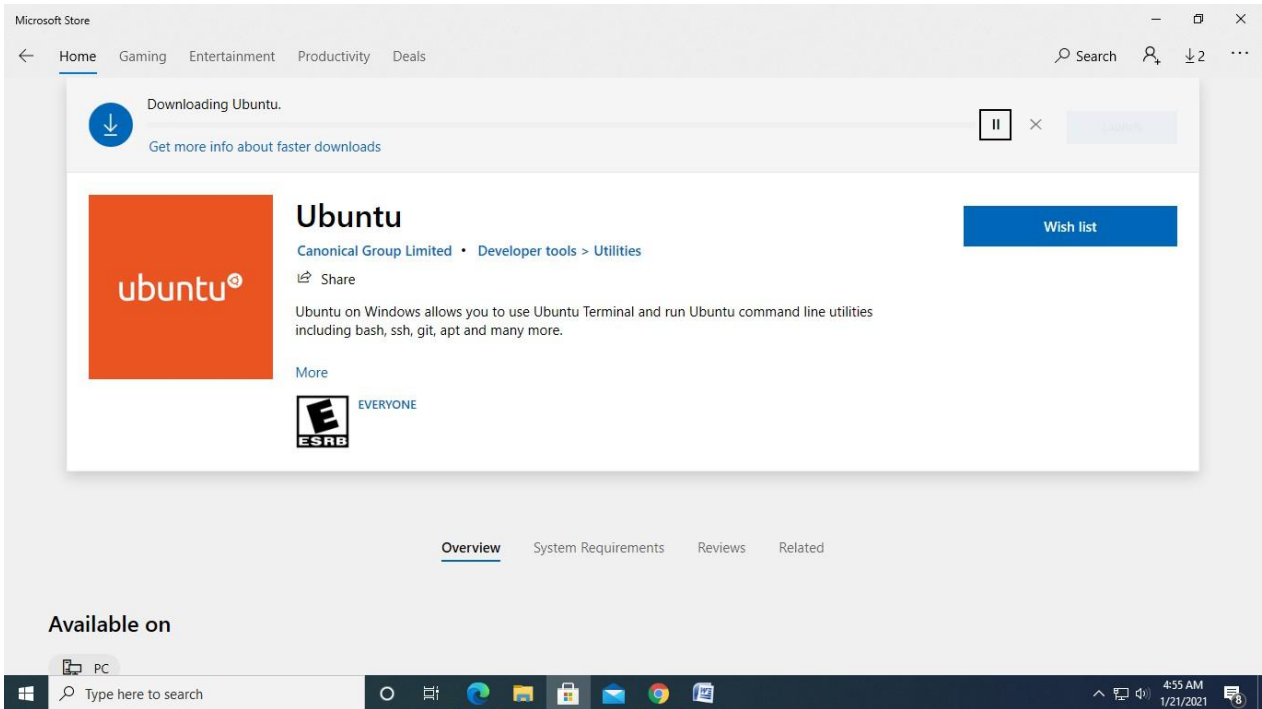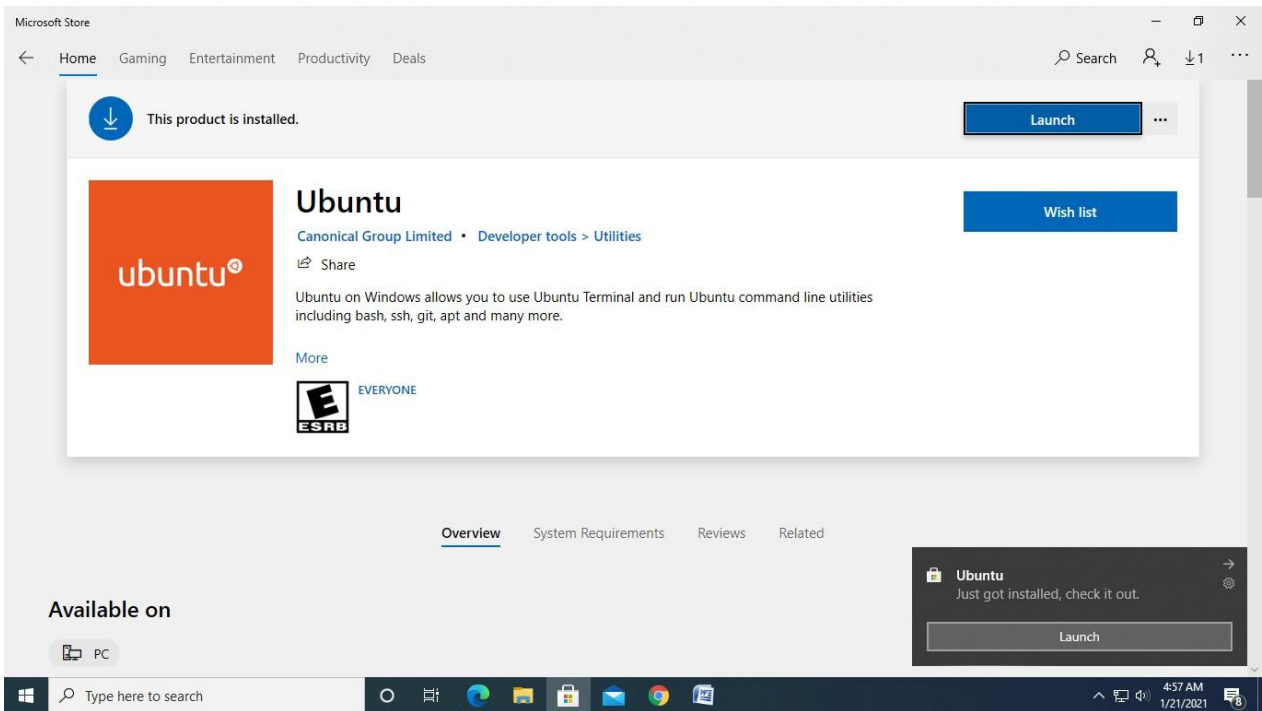
Now Launch the Ubuntu
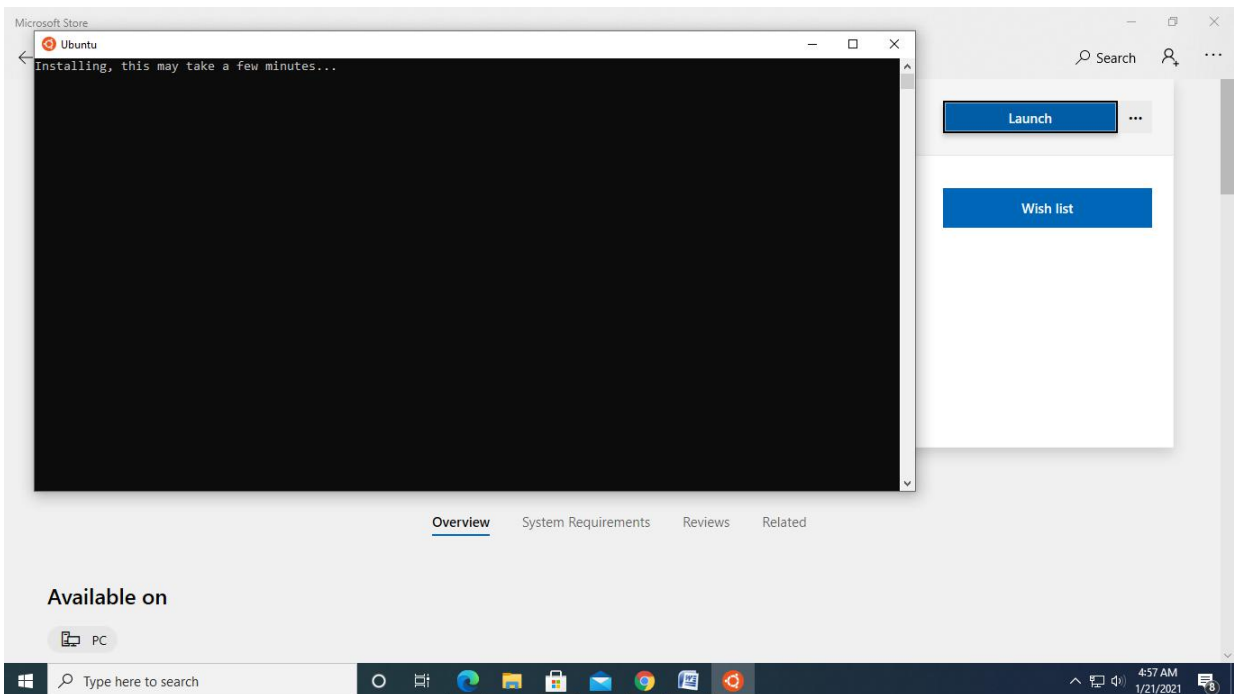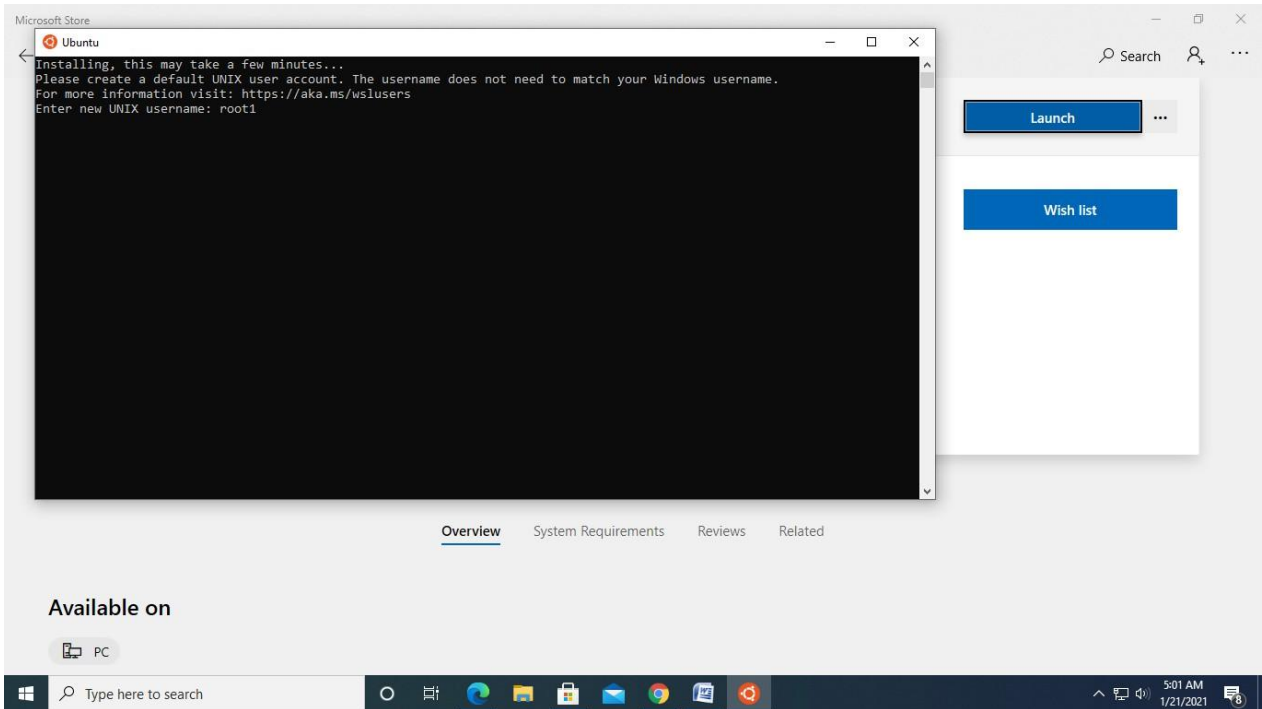
Username : Any meaningful

name Password: any name
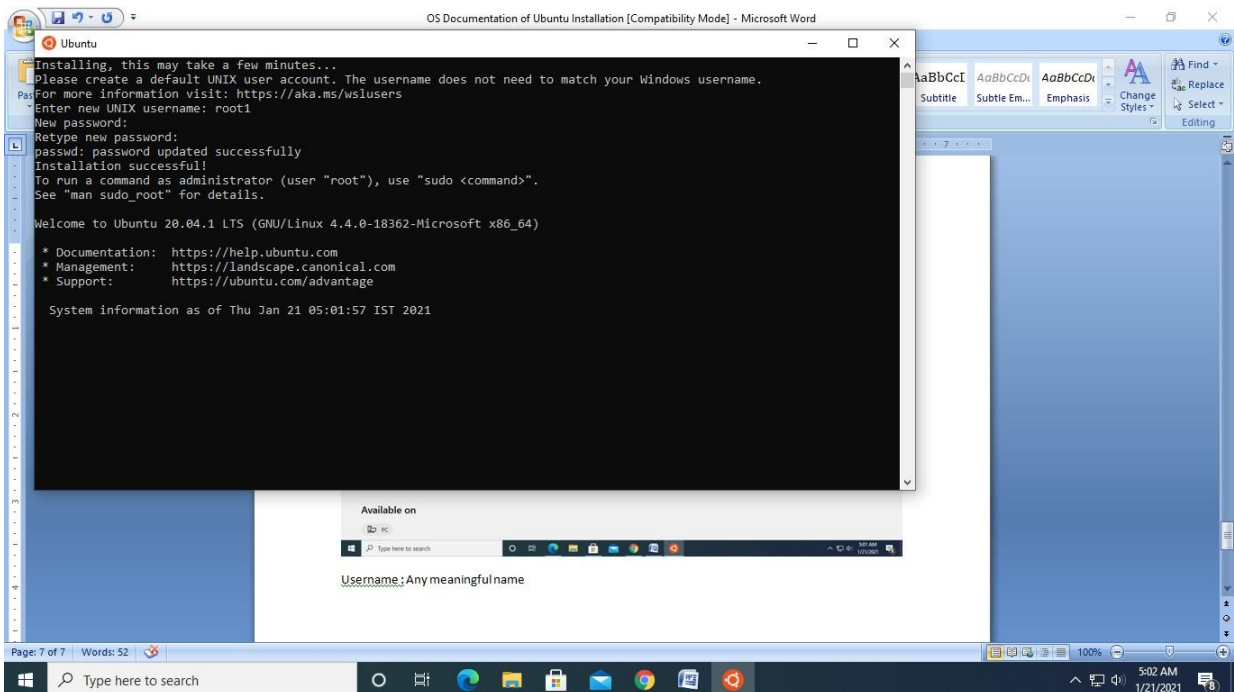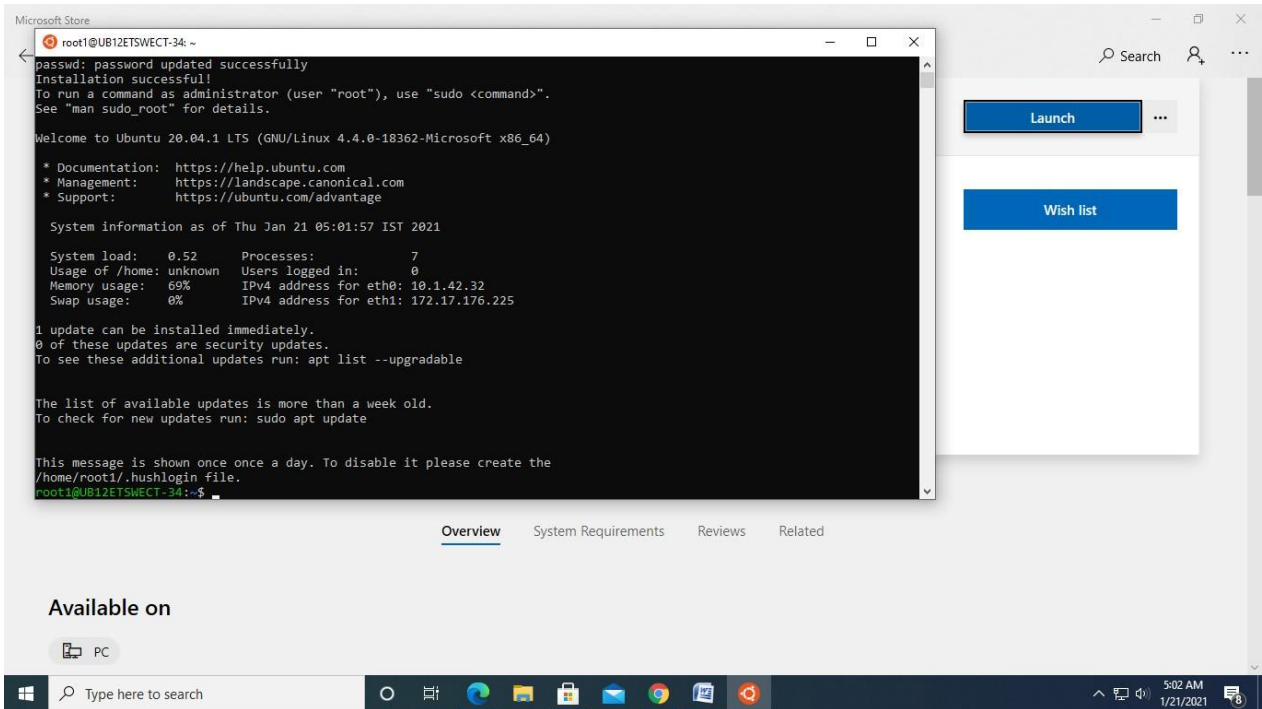
Finally you will get the prompt on successful installation of Ubuntu in windows 10 64 bit

LIST OF EXPERIMENTS

Ex.No.	Experiment Name
1	Understanding the booting process of Linux
2	Understanding the Linux file system
3	Understanding the various Phases of Compilation of a 'C' Program
4	System admin commands – Basics
5	System admin commands – Simple task automations
6	Linux commands
7	Shell Programs – Basic level
8	Process Creation
9	Overlay concept
10	Overlay concept
11	IPC using Pipes
12	IPC using shared memory and Message queues
13	Process synchronization
14	Study of OS161
15	Understanding the OS161 filesystem and working with test programs

Ex. No. 1	UNDERSTANDING THE BOOTING PROCESS OF LINUX

Objective:

To study various stages of Linux boot process.

Press the power button on your system, and after few moments you see the Linux login prompt. From the time you press the power button until the Linux login prompt appears, the following sequence occurs. The following are the 6 high level stages of a typical Linux boot process.

Power On	
BIOS	Basic Input/Output System executes MBR
MBR	Master Boot Record executes GRUB
GRUB	Grand Unified Bootloader executes Kernel
Kernel	Kernel executes /sbin/init
Init	Init executes runlevel programs
Runlevel	Runlevel programs are executed from /etc/rc.d/rc*.d/
	Login Process
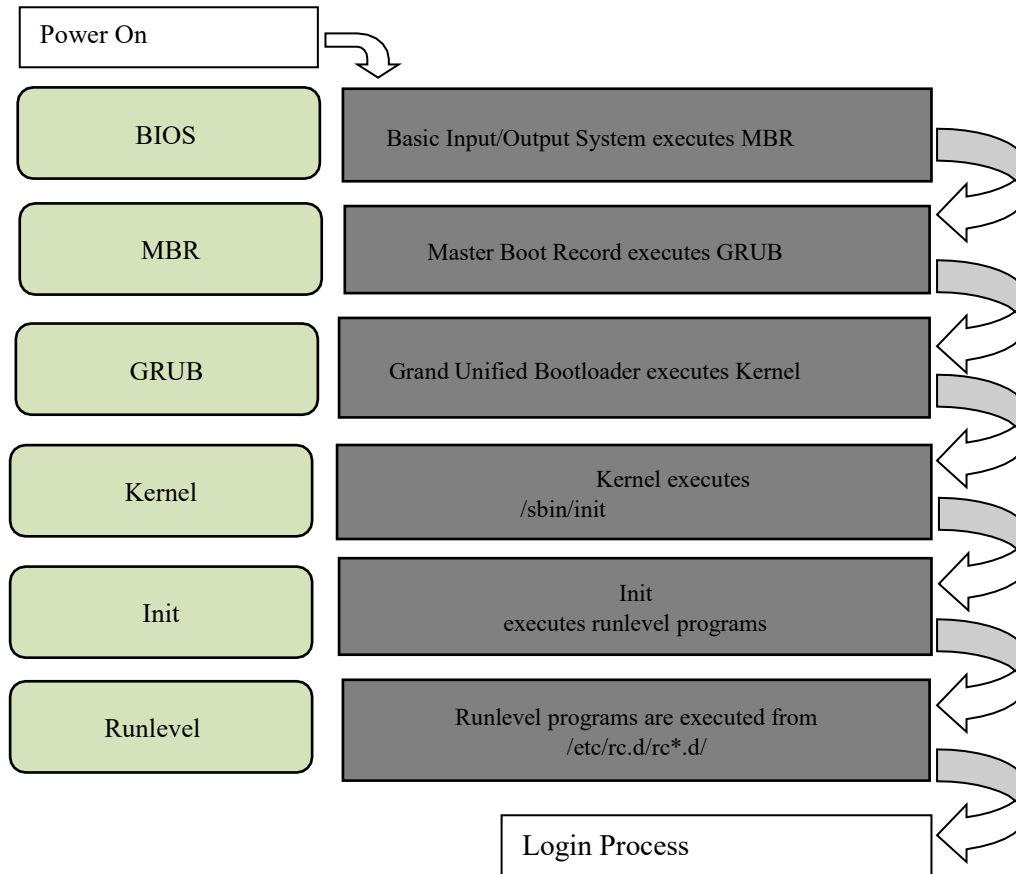
Step 1.BIOS

➢ BIOS stands for Basic Input/ Output System
➢ Performs some system integrity checks
➢ Searches, loads, and executes the boot loader program.
➢ It looks for boot loader in floppy, CD-ROMs, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence.
➢ Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
➢ So, in simple terms BIOS loads and executes the MBR boot loader.

Step 2. MBR

➢ MBR stands for Master Boot Record.
➢ It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
➢ MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
➢ It contains information about GRUB (or LILO in old systems).
➢ So, in simple terms MBR loads and executes the GRUB boot loader.

Step 3. GRUB

➢ GRUB stands for Grand Unified Bootloader.
➢ If you have multiple kernel images installed on your system, you can choose which one to be executed.
➢ GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
➢ GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
➢ Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this).
➢ As you notice from the above info, it contains kernel and initrd image.
➢ So, in simple terms GRUB just loads and executes Kernel and initrd images.

Step 4. Kernel

➢ Mounts the root file system as specified in the "root=" in grub.conf
➢ Kernel executes the /sbin/init program
➢ Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a 'ps -ef | grep init' and check the pid.
➢ initrd stands for Initial RAM Disk.
➢ initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

Step 5. Init

➢ Looks at the /etc/inittab file to decide the Linux run level.
➢ Following are the available run levels
- 0 – halt
- 1 – Single user mode
- 2 – Multiuser, without NFS
- 3 – Full multiuser mode
- 4 – unused
- 5 – X11
- 6 – reboot

➤ Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
➤ Execute 'grep initdefault /etc/inittab' on your system to identify the default run level
➤ If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
➤ Typically you would set the default run level to either 3 or 5.

Step 6. Runlevel programs

➤ When the Linux system is booting up, you might see various services getting started. For example, it might say "starting sendmail …. OK". Those are the runlevel programs, executed from the run level directory as defined by your run level.
➤ Depending on your default init level setting, the system will execute the programs from one of the following directories.
  o Run level 0 – /etc/rc.d/rc0.d/
  o Run level 1 – /etc/rc.d/rc1.d/
  o Run level 2 – /etc/rc.d/rc2.d/
  o Run level 3 – /etc/rc.d/rc3.d/
  o Run level 4 – /etc/rc.d/rc4.d/
  o Run level 5 – /etc/rc.d/rc5.d/
  o Run level 6 – /etc/rc.d/rc6.d/
➤ Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
➤ Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
➤ Programs starts with S are used during startup. S for startup.
➤ Programs starts with K are used during shutdown. K for kill.
➤ There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
➤ For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

Login Process

1. Users enter their username and password
2. The operating system confirms your name and password.
3. A "shell" is created for you based on your entry in the "/etc/passwd" file
4. You are "placed" in your "home"directory.
5. Start-up information is read from the file named "/etc/profile". This file is known as the system login file. When every user logs in, they read the information in this file.
6. Additional information is read from the file named ".profile" that is located in your "home" directory. This file is known as your personal login file.

Outcome:
      Learned the various stages of Linux boot process.

Ex. No. 2	UNDERSTANDING THE LINUX FILE SYSTEM

Objective:

To study various Linux file system and file system structure.

Linux File System

Linux File System or any file system generally is a layer which is under the operating system that handles the positioning of your data on the storage, without it; the system cannot knows which file starts from where and ends where.

Linux offers many file systems types like:

➢ **Ext**: an old one and no longer used due to limitations.

➢ **Ext2**: first Linux file system that allows 2 terabytes of data allowed.

➢ **Ext3**: came from Ext2, but with upgrades and backward compatibility.

➢ **Ext4**: faster and allow large files with significant speed. (Best Linux File System). It is a very good option for SSD disks and you notice when you try to install any Linux distro that this one is the default file system that Linux suggests.

➢ **JFS**: old file system made by IBM. It works very well with small and big files, but it failed and files corrupted after long time use, reports say.

➢ **XFS**: old file system and works slowly with small files.

➢ **Btrfs:** made by Oracle. It is not stable as Ext in some distros, but you can say that it is a replacement for it if you have to. It has a good performance.

➢ **Nfs:** The network file system used to access disks located on remote computers.

➢ **Ntfs:** replaces Microsoft Window's FAT file systems (VFAT, FAT32). It has reliability, performance, and space- utilization.

➢ **Umsdos: It** is an extended DOS file system used by Linux.

File System Structure

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired. UNIX uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

The following table provides a short overview of the most important higher-level directories you find on a Linux system

Directory	Contents
/	Root directory—the starting point of the directory tree.
/bin	Essential binary files. Binary Executable files
/boot	Static files of the boot loader.
/dev	Files needed to access host-specific devices.
/etc	Host-specific system configuration files.
/lib	Essential shared libraries and kernel modules.
/media	Mount points for removable media.
/mnt	Mount point for temporarily mounting a file system.
/opt	Add-on application software packages.
/root	Home directory for the super user root.
/sbin	Essential system binaries.
/srv	Data for services provided by the system.
/proc	Contains all processes marked as a file by process number or other information that is dynamic to the system
/tmp	Temporary files.
/usr	Secondary hierarchy with read-only data.
/var	Variable data such as log files
/kernal	Contains kernel files

EDITORS AND FILTERS

VI EDITOR
- vi fname → to open the file fname

- There are two types of mode in vi editor
  Escape mode – used to give commands – to switch to escape mode, press <Esc> key
  Command mode – used to edit the text – to switch to command mode, press any one the following inserting text command

a) Inserting Text

i	→ insert text before the cursor
a	→ append text after the cursor
I	→ insert text at the beginning of the line
A	→ append text to the end of the line
r	→ replace character under the cursor with the next character typed
R	→ Overwrite characters until the end of the line
o	→ (small o) open new line after the current line to type text
O	→ (capital O) open new line before the current line to type text

b) Cursor movements

h	→ left
j	→ down
k	→up
l	→ right

(The arrow keys usually work also)

^F	→forward one screen
^B	→back one screen
^D	→down half screen
^U	→up half screen

(^ indicates control key; case does not matter)

0 → (zero) beginning of line
$ → end of line

c) Deleting text
Note : (n) indicates a number, and is optional

dd	→ deletes current line
(n) **dd**	→ deletes (n) line(s) ex. 5dd → deletes 5 lines
(n)**dw**	→ deletes (n) word(s)
D	→ deletes from cursor to end of line

     x   → deletes current character
     (n)**x**   → deletes (n) character(s)
     X   → deletes previous character

d) Saving files
     :w   → to save & resume editing (write & resume)
     :wq   → to save & exit (write & quit)
     :q!   → quit without save

e) Cut, Copy and Paste
     yy   → copies current line
     (n) yy → copies (n) lines from the current line. ex. 4yy copies 4 lines.
     p → paste deleted or yanked (copied) lines after the cursor

FILTERS

1. cut
- Used to cut characters or fileds from a file/input

    Syntax :   **cut**   **-c**chars filename
                                 -ffieldnos filename

- By default, tab is the filed separator(delimiter). If the fileds of the files are separated by any other character, we need to specify explicitly by **–d** option

         cut   **-d**delimitchar **-f**fileds filname

2. paste
- Paste files vertically. That is n^{th} line of first file and n^{th} line of second file are pasted as the nth line of result

    Syntax :   **paste** file1 file2

-ddchar   option is used to paste the lines using the delimiting character *dchar*
-s   option is used paste the lines of the file in a single line

3. tr
- Used to translate characters from standard input

    Syntax :   **tr** char1 char2 < filename

    It translates char1 into char2 in file filename

- Octal representation characters can also be used

Octal value	Character
'\7'	Bell
'\10'	Backspace
'\11'	Tab

'\12'   Newline
'\33'   Escape

Ex.   tr : '\11' < f1   *translates all* **:** *into tab of f ile f1*

-s   Option translate multiple occurrences of a character by single character.
-d   Option is to delete a character

4. grep
▪ Used to search one or more files for a particular pattern.

Syntax :   **grep** pattern filename(s)

> Lines that contain the *pattern* in the file(s) get displayed
> pattern can be any regular expressions
> More than one files can be searched for a pattern

-v   option displays the lines that do not contain the *pattern*
-l   list only name of the files that contain the *pattern*
-n   displays also the line number along with the lines that matches the *pattern*

5. sort
▪ Used to sort the file in order

Syntax :   **sort** filename

> Sorts the data as text by default
> Sorts by the first filed by default

-r   option sorts the file in descending order
-u   eliminates duplicate lines
-o filename   writes sorted data into the file *fname*
-tdchar   sorts the file in which fileds are separated by *dchar*
-n   sorts the data as number
+1n   skip first filed and sort the file by second filed numerically

6. Uniq
▪ Displays unique lines of a sorted file
Syntax :   **uniq**   filename

-d   option displays only the duplicate lines
-c   displays unique lines with no. of occurrences.

7. cmp
▪ Used to compare two files
Syntax :   **cmp** f1 f2
               compare two files f1 & f2 and prints the line of first difference .

8. diff
- Used to differentiate two files

  Syntax :   **diff** f1 f2
             compare two files f1 & f2 and prints all the lines that are differed between f1 & f2.

9. comm
- Used to compare two sorted files
  Syntax :   **comm** file1 file2

             Three columns of output will be displayed.
             First column displays the lines that are unique to file1
             Second column displays the lines that are unique to file2
             Third column displays the lines that are appears in both the files

  -1   option suppress first column
  -2   option suppress second column
  -3   option suppress third column
  -12   option display only third column
  -13   option display only second column
  -23   option display only first column

Outcome:
           Learned and used various Linux file system, file system structure, VI editor and various filter commands in Linux and executed.

Ex. No. 3	UNDERSTANDING THE VARIOUS PHASES OF COMPILATION OF C PROGRAM

Objective:

          To practice how to create and execute C Programs in Linux.

Compilation of C Program

Step 1: Open the terminal and edit your program using vi editor/gedit editor and save with extension ".c"
          Ex.   vi test.c
              (or)
                gedit text.c

Step 2: Compile your program using gcc compiler
          Ex.   gcc test.c → Output file will be "a.out"
              (or)
              gcc –o test text.c → Output file will be "test"

Step 3: Correct the errors if any and run the program
          Ex.   ./a.out
              or
              ./test

Optional Step: In order to avoid. / prefix each time a program is to be executed, insert the following as the last line in the file **.profile**
          export PATH=.:$PATH
This Step needs only to be done once.

Debug C Programs using gdb debugger

Step 1 : Compile C program with debugging option –g
    Ex.   gcc –g test.c

Step 2 : Launch gdb. You will get gdb prompt
    Ex.   gdb a.out

Step 3 : Step break points inside C program
    Ex.   (gdb) b 10
    Break points set up at line number 10. We can have any number of break points

Step 4: Run the program inside gdb
    Ex. (gdb) r

Step 5: Print variable to get the intermediate values of the variables at break point
    Ex.   (gdb) p i   → Prints the value of the variable 'i'

Step 6: Continue or stepping over the program using the following gdb commands

    c → continue till the next break
    n → Execute the next line. Treats function as single statement
    s → Similar to 'n' but executes function statements line by line
    l → List the program statements

Step 7: Quit the debugger
    (gdb) q

Outcome:
    Learned and practiced how to create and execute C Programs in Linux.