EX NO: 1

DATE:

Installation of windows operating system

AIM :

Installing Windows7 Operating System:

System Requirements

If you are currently running Windows Vista, the good news is that you are ready for Windows 7. Tests performed by various parties have consistently confirmed that Windows 7 outperformed Windows Vista on a similar hardware configuration.

If you are coming from previous versions of Windows (pre-Vista), take note of the following *suggested* hardware requirements:

- 1 GHz or faster 32-bit or 64-bit processor
- 1 GB RAM (for 32-bit) or 2 GB RAM (for 64-bit)
- 16 GB of available disk space (for 32-bit) or 20 GB of available disk space (for 64-bit)
- DirectX 9 graphics device with Windows Display Driver Model 1.0 or higher (for Aero—the graphical user interface and default theme in most editions of Windows 7)

Installing Windows 7

If you are currently running Windows Vista (with Service Pack 1), you can upgrade to Windows 7 directly from within Vista. Windows XP users will need to install a fresh copy of Windows 7.

The following steps will walk you through the process of installing Windows 7 on a fresh computer.

Installing Windows 7 is straightforward—if you're doing a clean install, simply boot up your computer with the Windows 7 installation DVD inside the DVD drive and instruct your computer to boot from the DVD (you may need to press a key, such as F11 or F12, while the computer is starting to enter the boot selection screen). If you're upgrading, simply boot into Windows Vista, insert the disc, and run the installer (if you are using Windows XP, see the previous sidebar).

When the installer has booted up, you will be greeted with the screen shown in Figure 1(the upgrade screen is slightly different; you will have an option to check the compatibility of your system or start the installation). You will be asked to select the language to install, the time and currency format, and your keyboard type

EX NO:	
DATE:	

Fig 1:Installing Windows 7: the first step

With the selections made, you can now install Windows 7 by clicking the “Install now” button (see Figure 1-3).

You will be asked to accept the license agreement. (If you are upgrading, you’ll first have the option to go online to get any updates to the installer first.) Check the licensing checkbox and continue.

EX NO:	
DATE:	

Fig.2 .Click the “Install now” button to start the Windows 7 installation process

On the next screen, you have a choice between upgrading your existing Windows or installing a fresh copy of Windows. If you are using Windows XP or earlier, the first option will not work for you—select the Custom (advanced) option

On the next screen, you have a choice between upgrading your existing Windows or installing a fresh copy of Windows. If you are using Windows XP or earlier, the first option will not work for you—select the Custom (advanced) option

For a fresh installation, you will be asked to select a disk for installing Windows 7. Select the appropriate disk and click Next (see Figure 3). If you are upgrading, the Windows 7 installer will generate a compatibility report and save it to your desktop.

EX NO:	
DATE:	

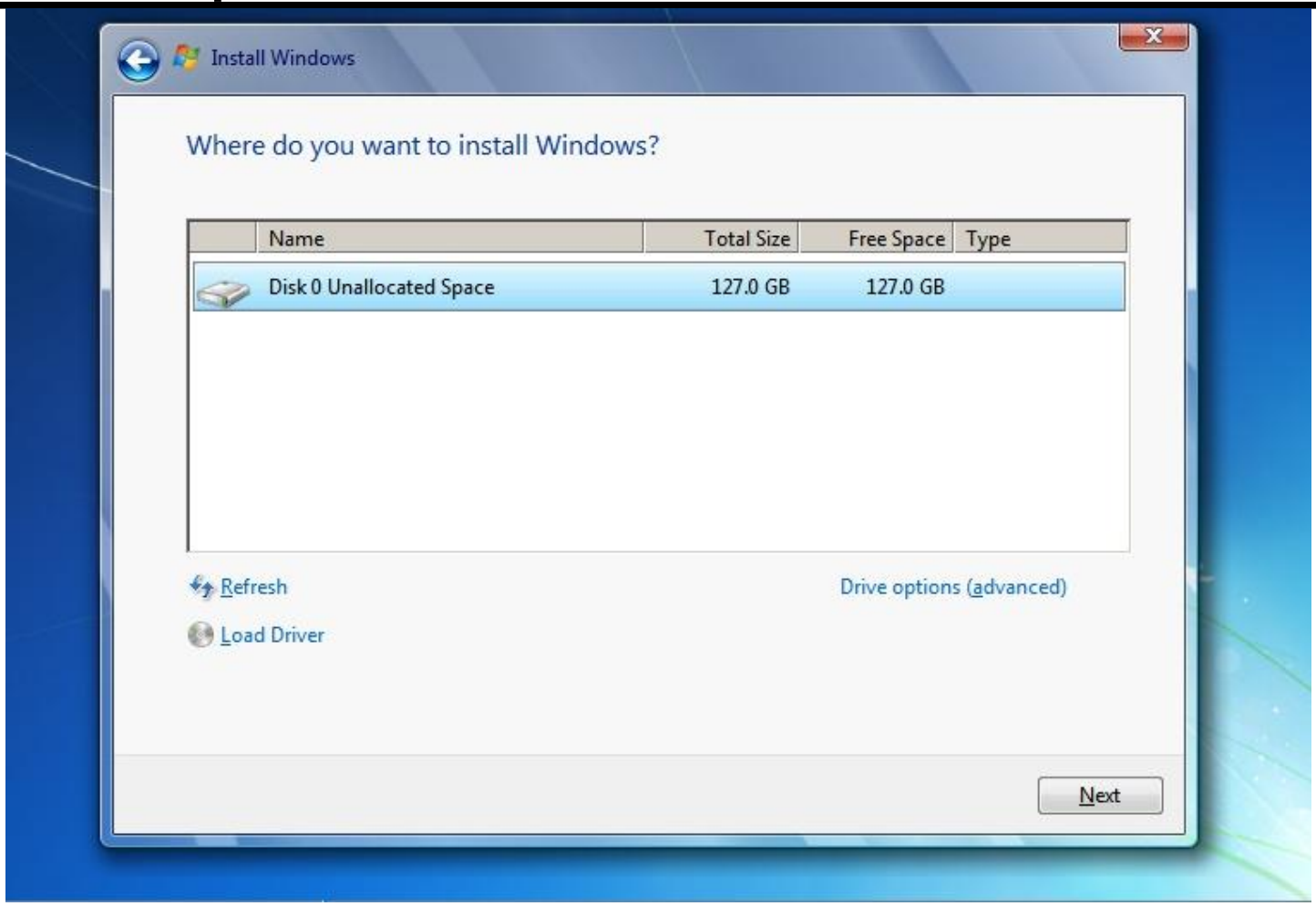


Fig.3. Selecting the disk for installing Windows 7

Windows will now take some time to copy all the files into the selected disk and proceed with the installation (see Figure 4). This will take about 20–30 minutes, depending on the speed of your computer.

EX NO:	
DATE:	

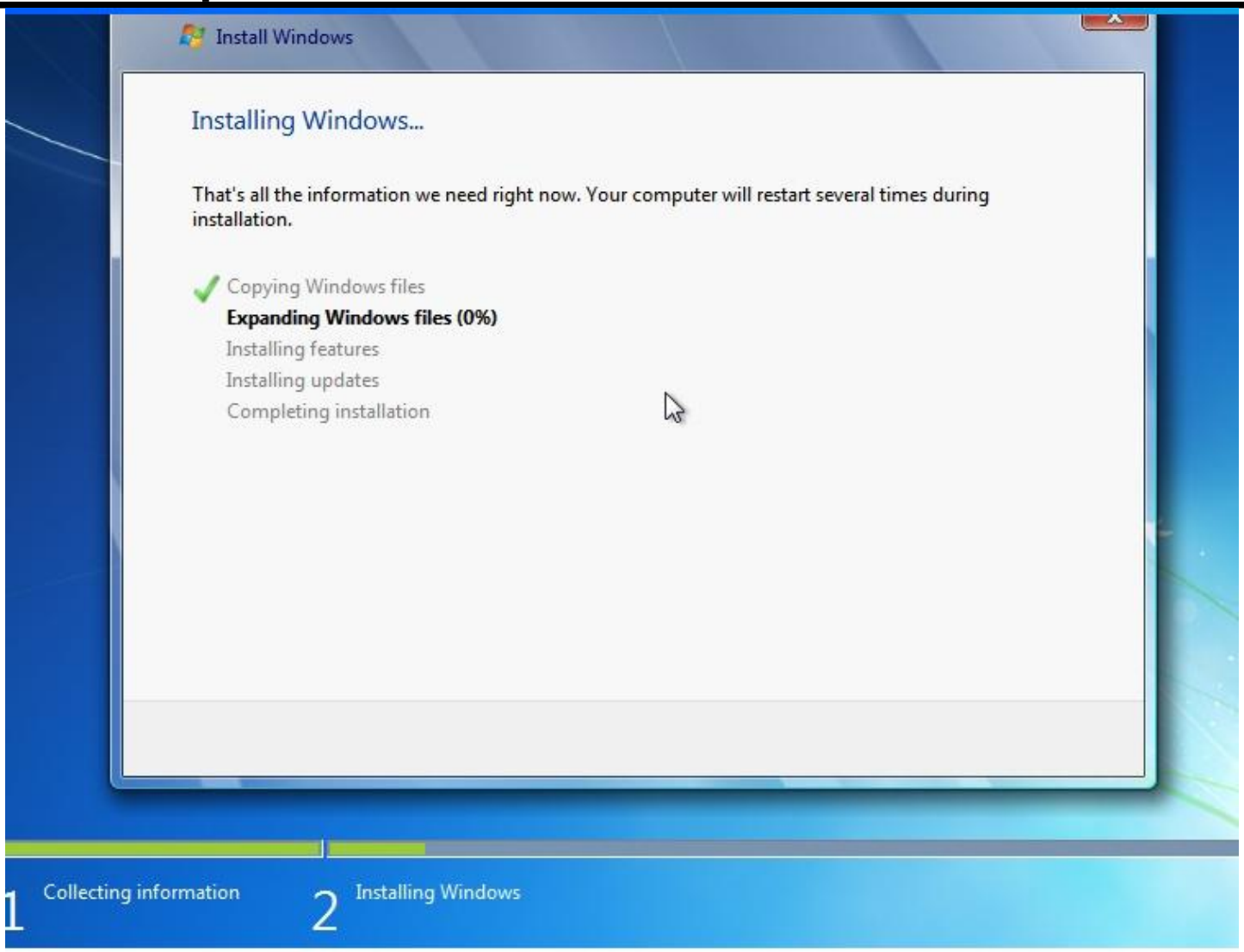


Fig 4 . Windows 7 proceeding with the installation

When the installation is complete, Windows 7 will restart. After Windows 7 has been restarted, you should see the screen shown in Figure 1-7. Provide a username; your computer name will be created based on what you have entered (you can change it to another name if you want to after the installation). Click Next.

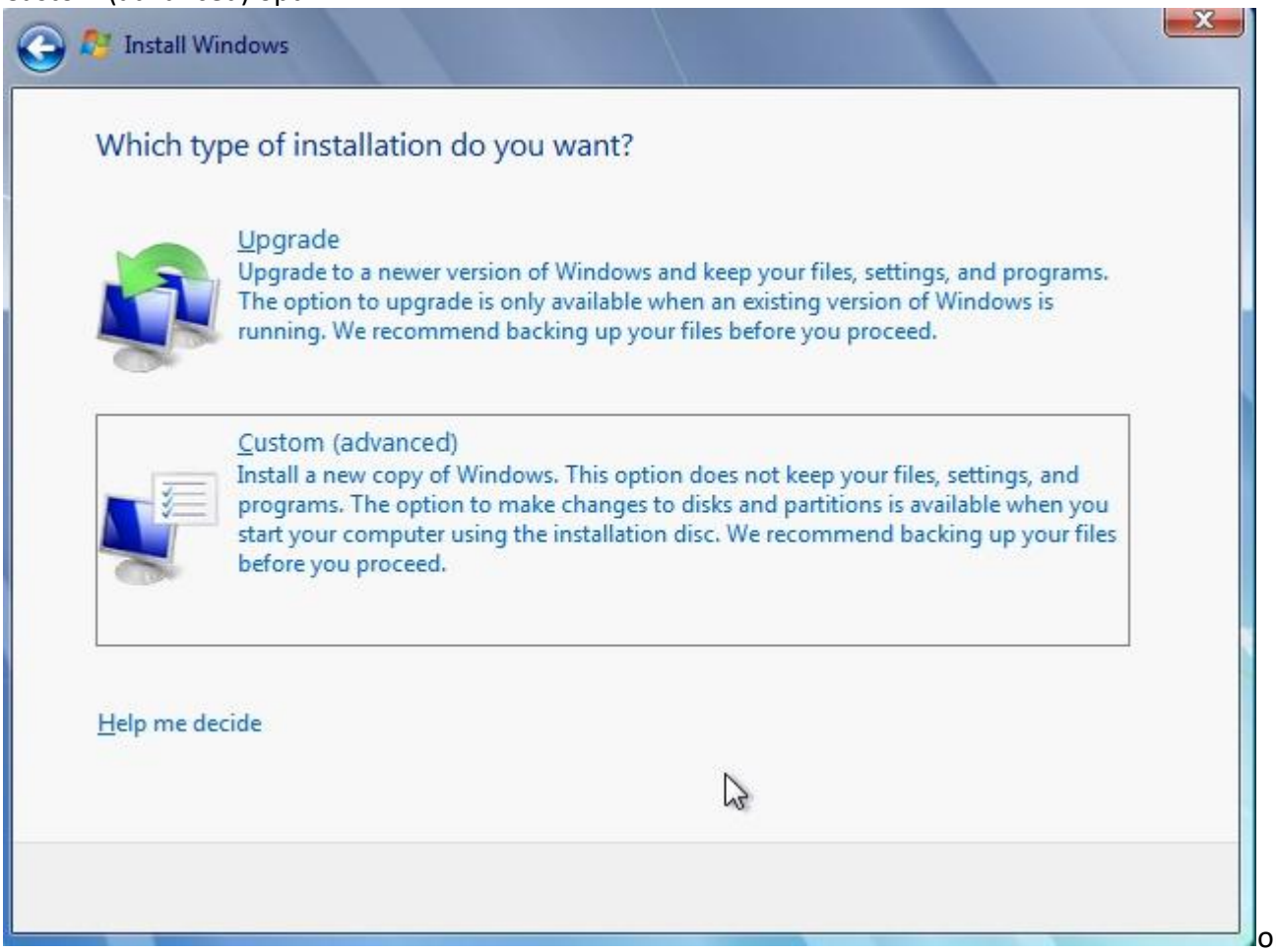
EX NO:	
DATE:	

Fig 5 .Click the “Install now” button to start the Windows 7 installation process

On the next screen, you have a choice between upgrading your existing Windows or installing a fresh copy of Windows. If you are using Windows XP or earlier, the first option will not work for you—select the

EX NO:	
DATE:	

Custom (advanced) opti



on

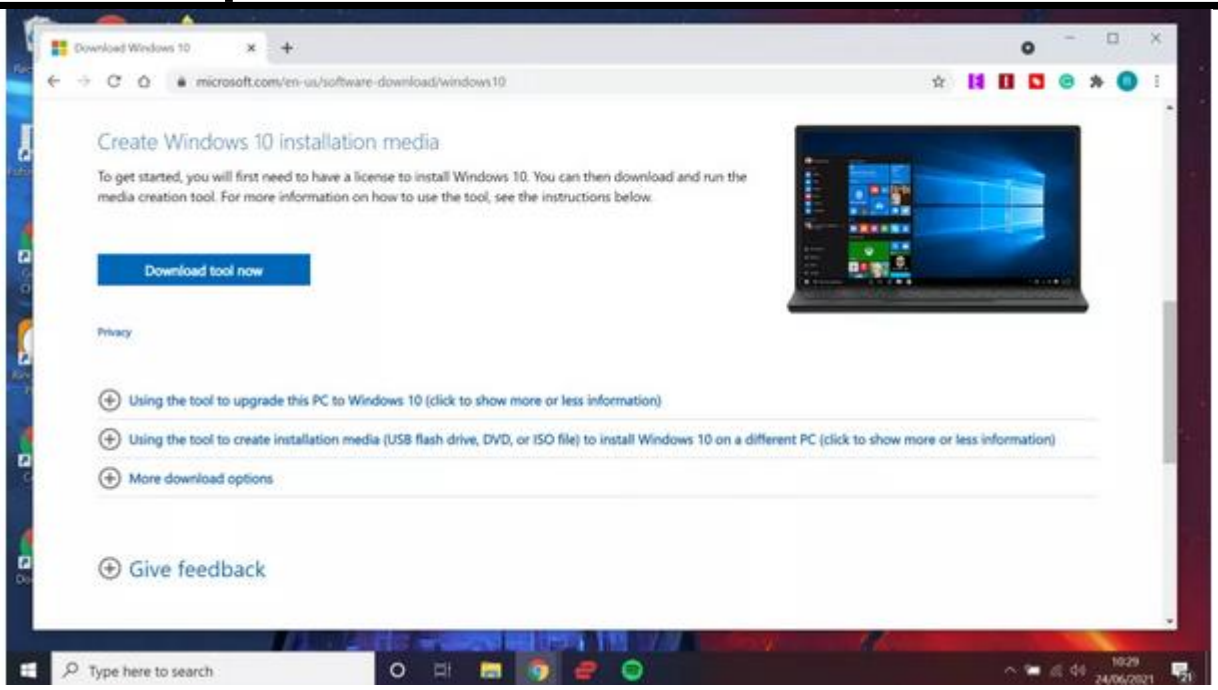
Installing Windows 10 Operating System:

1. Check your device meets the Windows 10 system requirements. Below you'll find the minimum specs needed to run Windows 10, so check your device is capable:

CPU: 1GHz or faster processor
RAM: 1GB for Windows 10 32-bit or 2GB for Windows 10 64-bit
Storage: 32GB of space or more
GPU: DirectX 9 compatible or later with WDDM 1.0 driver
Display: 800x600 resolution or higher

2. Create USB installation media. Visit Microsoft's Windows 10 download page (opens in new tab) and select "Download tool now" under the "create Windows 10 installation media" section. Transfer the downloaded installer tool to a USB drive.

EX NO:	
DATE:	



3. Run the installer tool. Open the installer tool by clicking on it. Accept Microsoft's terms, and then **select "Create installation media for another PC"** on the "What do you want to do?" page. After selecting which language you want Windows 10 to run in, and which edition you want as well (32-bit or 62-bit), you'll be asked what type of media you want to use.

Installing from a USB drive is definitely the preferred option but you can also install from a CD or ISO file. Once you choose your device, the installer tool will download the required files and put them onto your drive.

4. Use your installation media. Insert your installation media into your device and then **access the computer's BIOS or UEFI**. These are the systems that allow you to control your computer's core hardware.

The process of accessing these systems is unique to each device, but the manufacturer's website should be able to give you a helping hand here. Generally, you'll need to **press the F2, F12 or Delete keys** as your computer boots up.

5. Change your computer's boot order. Once you have access to your computer's BIOS/UEFI you'll need to locate the settings for boot order. You need the Windows 10 installation tool to be higher up on the list than the device's current current boot drive: this is the SSD or HDD that your existing OS is stored on. You should **move the drive with the installer files to the very top of the boot order menu**. Now, when you restart your device the Windows 10 installer should load up first.

6. Restart your device. Save your settings in the BIOS/UEFI and reboot your device.

7. Complete the installation. Your device should now load up the Windows 10 installation tool on restart. This will guide you through the rest of the installation process.

RESULT:

EX NO: 2

DATE:

Illustration of UNIX commands and Shell Programming

2.1. BASICS OF UNIX COMMANDS

AIM :

COMMAND:

1. Date Command:

This command is used to display the current data and time.

Syntax :

$date -options

Options : -

a = Abbrevated weekday.
A = Full weekday.
b = Abbrevated month.
B = Full month.
c = Current day and time.
C = Display the century as a decimal number.
d = Day of the month.
D = Day in „mm/dd/yy" format
h = Abbrevated month day.
H = Display the hour.
L = Day of the year.
m = Month of the year.
M = Minute.
P = Display AM or PM
S = Seconds
T = HH:MM:SS format
u = Week of the year.
y = Display the year in 2 digit.
Y = Display the full year.
Z = Time zone .
To change the format :

Syntax :

$date "+%H-%M-%S"

Output:

16-24-19

2. Calender Command:

This command is used to display the calendar of the year or the particular month of calendar year.

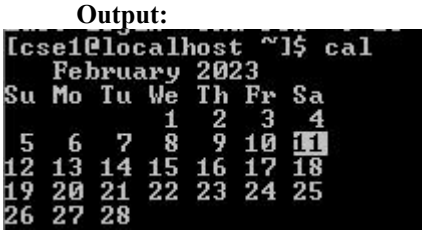
Syntax :

a.$cal <year>

b.$cal <month><year>

Here the first syntax gives the entire calendar for given year & the second Syntax gives the calendar of reserved month of that year.

Output:



3. Echo Command:

This command is used to print the arguments on the screen.

EX NO:	
DATE:	

Syntax : $echo <text>

Multi line echo command:

To have the output in the same line, the following commands can be used.

Syntax : $echo text

To have the output in different line, the following command can be used.

Syntax : $echo "text

>line2

>line3"

Input &Output:

```
[cse35@localhost ~]$ echo "Hello
> how
> r
> u"
Hello
how
r
u
```

4.'who' Command :

It is used to display who are the users connected to our computer currently.

Syntax : $who –option's

Options : -

H–Display the output with headers.

b–Display the last booting date or time or when the system was lastely rebooted.

Input &Output:

```
[cse35@localhost ~]$ who
cse35    pts/1        2017-12-28 16:06 (172.16.10.123)
[cse35@localhost ~]$ who -H
NAME     LINE         TIME             COMMENT
cse35    pts/1        2017-12-28 16:06 (172.16.10.123)
[cse35@localhost ~]$ who -b
system boot  2017-12-28 16:03
```

5.'who am i' Command :

Display the details of the current working directory.

Syntax : $who am i

Input &Output:

```
[cse35@localhost ~]$ who am i
cse35    pts/1        2017-12-28 16:06 (172.16.10.123)
```

6.'tty' Command :

It will display the terminal name.

Syntax : $tty

```
[cse1@localhost ~]$ tty
/dev/pts/1
```

7.'Binary' Calculator Command:

Use *bc* is to enter the calculator's own shell.We can run many calculations in a row.

$ bc

bc 1.06.95

Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.

This is free software with ABSOLUTELY NO WARRANTY.

For details type `warranty'.

Now we can type in calculations or commands, one per line:

1+1

OUTPUT:

2

Input2:

(4+7)*2

4+7*2

EX NO:	
DATE:	

8.'CLEAR' Command:

It is used to clear the screen.

Syntax : $clear

9.'MAN' Command:

It helps us to know about the particular command and its options & working. It is like "help command in windows.

Syntax : $man <command name>

10. tput Command:

The **tput** command uses the **terminfo** database to make terminal-dependent information available to the shell. The **tput** command outputs a string if the attribute *CapabilityName* is of type *string*. The output string is an integer if the attribute is of type *integer*. If the attribute is of type *Boolean*, the **tput** command sets the exit value (0 for TRUE, 1 for FALSE), and produces no other output.

Syntax : $tput <argument>

XTERM DESCRIPTION LIMITATION

The xterm terminal description in the DEC.TI file on AIX® Version 4 provides underline mode by using the SGR attribute. The SMUL and RMUL attributes are not currently defined in the XTERM terminal description on AIX Version 4. Use the more generic capability named SGR("**Select Graphic Rendition**", which is also known as "Set Attribute".)

tput sgr x y

Where *x* is either a 1 or a 0 to turn standout mode on or off respectively, and *y* is either a 1 or a 0 to turn underline mode on or off respectively. See the article "**terminfo** file format" for more details on the SGR capability.

tput sgr 0 1 turn off standout; turn on underline
tput sgr 0 0 turn off standout; turn off underline
tput sgr 1 1 turn on standout; turn on underline
tput sgr 1 0 turn on standout; turn off underline

.

Output:

11. LIST Command:

It is used to list all the contents in the current working directory.

Syntax: $ ls –options <arguments>

If the command does not contain any argument means it is working in the Current directory.

Options:

a– used to list all the files including the hidden files.
c– list all the files columnwise.
d- list all the directories.
m- list the files separated by commas.
p- list files include "/" to all the directories.
r- list the files in reverse alphabetical order.
f- list the files based on the list modification date.
x-list in column wise sorted order.

Input &Output:

```
[cse35@localhost ~]$ ls
aaa  ff  fff
[cse35@localhost ~]$ ls -a
.  aaa          .bash_logout   .bashrc  ff   .gnome2
.. .bash_history .bash_profile .emacs   fff  .zshrc
[cse35@localhost ~]$ ls -c
ff  fff  aaa
[cse35@localhost ~]$ ls -d
.
[cse35@localhost ~]$ ls -m
aaa, ff, fff
[cse35@localhost ~]$ ls -p
```

EX NO:	
DATE:	

aaa/ ff fff/
[cse35@localhost ~]$ ls -r
fff ff aaa
[cse35@localhost ~]$ ls -f
.emacs .bash_profile ff .bash_logout fff .gnome2
.bashrc .. . aaa .bash_history .zshrc
[cse35@localhost ~]$ ls -x
aaa ff fff
[cse35@localhost ~]$

DIRECTORY RELATED COMMANDS:

1. **Present Working Directory Command :**

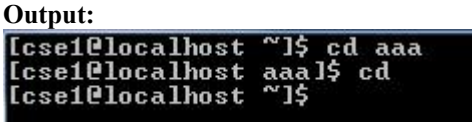
To print the complete path of the current working directory.
Syntax : $pwd
Input &Output:
[cse35@localhost ~]$ pwd
/home/cse35

2. **MKDIR Command :**

To create or make a new directory in a current directory.
Syntax : $mkdir <directory name>
Input &Output:
[cse35@localhost ~]$ mkdir folder1

3. **CD Command :**

To change or move the directory to the mentioned directory.
Syntax : $cd <directory name.
Output:



4. **RMDIR Command :**

To remove a directory in the current directory & not the current directory itself.
Syntax : $rmdir <directory name>

FILE RELATED COMMANDS:

1. CREATE A FILE:

To create a new file in the current directory we use CAT command.
Syntax :
$cat > filename
The > symbol is redirectory we use cat command.
Type '**Ctrl-d'** at the end to save the file.

2. DISPLAY A FILE:

To display the content of file mentioned we use CAT command without ">" **operator.**
Syntax :
$cat filename
Input &Output:
[cse35@localhost ~]$
[cse35@localhost ~]$ cat >example2
hi
hello
this is cat command file
[cse35@localhost ~]$ cat example2
hi
hello
this is cat command file

EX NO:	
DATE:	

3. COPYING CONTENTS:

To copy the content of one file with another. If file does not exist, a new file is created and if the file exists with some data then it is overwritten.

Syntax :

$ cat <filename source>>><destination filename>

$ cat <source filename>>><destination filename> it is avoid overwriting.

Options : -

-n content of file with numbers included with blank lines.

Input &Output:

[cse35@localhost ~]$ cat example2>>example3

[cse35@localhost ~]$ cat example3

hi

hello

this is cat command file

[cse35@localhost ~]$ cat example2

hi

hello

this is cat command file

4. SORTING A FILE:

To sort the contents in alphabetical order in reverse order.

Syntax :

$sort <filename >

Input &Output:

[cse35@localhost ~]$ sort example2

hello

hi

this is cat command file

5. COPYING CONTENTS FROM ONE FILE TO ANOTHER:

To copy the contents from source to destination file. So that both contents are same.

Syntax :

$cp <source filename><destination filename>

Output:

[cse35@localhost ~]$ cat example2

hi

hello

this is cat command file

[cse35@localhost ~]$ cp example2 example4

[cse35@localhost ~]$ cat example4

hi

hello

this is cat command file

[cse35@localhost ~]$

6. MOVE Command:

To completely move the contents from source file to destination file and to remove the source file.

Syntax :

$ mv <source filename> <destination filename>

Output:

[cse35@localhost ~]$ cat >yy.txt

This is my first file

created for copy content

thankyou

[cse35@localhost ~]$ cat yy.txt

This is my first file

created for copy content

thankyou

[cse35@localhost ~]$ mv yy.txt xx.txt

EX NO:	
DATE:	

[cse35@localhost ~]$ cat yy.txt
cat: yy.txt: No such file or directory
[cse35@localhost ~]$ cat xx.txt
This is my first file
created for copy content
thankyou

7. REMOVE Command:

Use the **rm command** to remove files you no longer need. The rm command removes the entries for a specified file, group of files, or certain select files from a list within a directory. User confirmation, read permission, and write permission are not required before a file is removed when you use the rm command.

Syntax :

$rm <filename>

8. WORD Command:

To list the content count of no of lines, words, characters.

Syntax :

$wc <filename>

Options :

-c – to display no of characters.
-l – to display only the lines.
-w – to display the no of words.
Input &Output:
[cse35@localhost ~]$ cat xx.txt
This is my first file
created for copy content
thankyou
[cse35@localhost ~]$ wc xx.txt
3 10 56 xx.txt

9. LINE PRINTER:

submits files for printing or alters a pending job.

Syntax :

$lp <filename>

10. PAGE Command:

This command is used to display the contents of the file page wise & next page can be viewed by pressing the enter key.

Syntax :

$pg <filename>

11. FILTERS AND PIPES

HEAD: It is used to display the top ten lines of file.

Syntax: $head <filename>

TAIL: This command is used to display the last ten lines of file.

Syntax: $tail <filename>

PAGE: This command shows the page by page a screen full of information is displayed after which the page command displays a prompt and passes for the user to strike the enter key to continue scrolling.

Syntax: $ls –a\p

MORE: It also displays the file page by page .To continue scrolling with more command, press the space bar key.

Syntax: $more <filename>

GREP: The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).

Syntax:

grep [options] pattern [files]

Options Description

-c : This prints only a count of the lines that match a pattern
-h : Display the matched lines, but do not display the filenames.

EX NO:	
DATE:	

-i : Ignores, case for matching
-l : Displays list of a filenames only.
-n : Display the matched lines and their line numbers.
-v : This prints out all the lines that do not matches the pattern
-e exp : Specifies expression with this option. Can use multiple times.
-f file : Takes patterns from file, one per line.
-E : Treats pattern as an extended regular expression (ERE)
-w : Match whole word
-o : Print only the matched parts of a matching line,
with each such part on a separate output line.

-A n : Prints searched line and nlines after the result.
-B n : Prints searched line and n line before the result.
-C n : Prints searched line and n lines after before the result.

$cat > geekfile.txt

unix is great os. unix was developed in Bell labs.
learn operating system.
Unix linux which one you choose.
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

$grep -i "UNix" geekfile.txt

Output:

unix is great os. unix was developed in Bell labs.
Unix linux which one you choose.
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

Case insensitive search : The -i option enables to search for a string case insensitively in the given file. It matches the words like “UNIX”, “Unix”, “unix”.

PIPE: It is a mechanism by which the output of one command can be channeled into the input of another command.
A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing.

Syntax: $who | wc -l

TR: The tr filter is used to translate one set of characters from the standard inputs to another.

Syntax: $tr “[a-z]” “[A-Z]”

WC: Word Command.Used to list the no of lines,words,characters

Syntax: $wc <filename>

COMMUNICATION THROUGH UNIX COMMANDS

1. MESG

Description: Mesg controls the access to your terminal by others. It’s typically used to allow or disallow other users to write to your terminal (see **write**(1)).

OPTIONS

EX NO:

DATE:

Tag	Description
y	Allow write access to your terminal.
n	Disallow write access to your terminal.

If no option is given, **mesg** prints out the current access state of your terminal.

Syntax: $mesg y

Input & Output:

[cse35@localhost ~]$ mesg

2. Command: WRITE

Description: This command is used to communicate with other users, who are logged in at the same time.

Syntax: $write <user name>

3. Command: WALL

Description: This command sends message to all users those who are logged in using the unix server.

Syntax: $wall <message>

4. Command: MAIL

Description: It refers to textual information, which can be transferred from one user to another

Syntax: $mail <user name>

5. Command: REPLY

Description: It is used to send reply to specified user.

Syntax: $reply <user name>

RESULT:

EX NO: 2.B

DATE:

SHELL PROGRAMMING
Study of Unix shell Programming

AIM:

INTRODUCTION :

Shell programming is a group of commands grouped together under single filename. After logging onto the system a prompt for input appears which is generated by a Command String Interpreter program called the shell. The shell interprets the input, takes appropriate action, and finally prompts for more input. The shell can be used either interactively - enter commands at the command prompt, or as an interpreter to execute a shell script. Shell scripts are dynamically interpreted, NOT compiled.

Common Shells.

C-Shell - csh : The default on teaching systems Good for interactive systems Inferior programmable features

Bourne Shell - bsh or sh - also restricted shell - bsh : Sophisticated pattern matching and file name substitution

Korn Shell : Backwards compatible with Bourne Shell Regular expression substitution emacs editing mode

Thomas C-Shell - tcsh : Based on C-Shell Additional ability to use emacs to edit the command line Word completion & spelling correction Identifying your shell.

01. SHELL KEYWORDS :

echo, read, if fi, else, case, esac, for , while , do , done, until , set, unset, readonly, shift, export, break, continue, exit, return, trap , wait, eval ,exec, ulimit , umask.

02. General things SHELL

The shbang line The "shbang" line is the very first line of the script and lets the kernel know what shell will be interpreting the lines in the script. The shbang line consists of a #! followed by the full pathname to the shell, and can be followed by options to control the behavior of the shell.

E*XAMPLE*

#!/bin/sh

Comments Comments are descriptive material preceded by a # sign. They are in effect until the end of a line and can be started anywhere on the line.

E*XAMPLE*

this text is not # interpreted by the shell

Wildcards There are some characters that are evaluated by the shell in a special way. They are called shell metacharacters or "wildcards." These characters are neither numbers nor letters. For example, the *, ?, and [ ] are used for filename expansion. The <, >, 2>, >>, and | symbols are used for standard I/O redirection and pipes. To prevent these characters from being interpreted by the shell they must be quoted.

E*XAMPLE*

Filename expansion:

rm *; ls ??; cat file[1-3];

Quotes protect metacharacter:

echo "How are you?"

03. SHELL VARIABLES :

EX NO:	
DATE:	

Shell variables change during the execution of the program .The C Shell offers a command "Set" to assign a value to a variable.

For example:

% set myname= Fred

% set myname = "Fred Bloggs"

% set age=20

A $ sign operator is used to recall the variable values.

For example:

% echo $myname will display Fred Bloggs on the screen

A @ sign can be used to assign the integer constant values.

For example:

%@myage=20

%@age1=10

%@age2=20

%@age=$age1+$age2

%echo $age

List variables

% set programming_languages= (C LISP)

% echo $programming _languages

C LISP

% set files=*.*

% set colors=(red blue green)

% echo $colors[2]

blue

% set colors=($colors yellow)/add to list

Local variables Local variables are in scope for the current shell. When a script ends, they are no longer available; i.e., they go out of scope. Local variables are set and assigned values.

EXAMPLE

variable_name=value name="John Doe" x=5

Global variables Global variables are called environment variables. They are set for the currently running shell and any process spawned from that shell. They go out of scope when the script ends.

EXAMPLE

VARIABLE_NAME=value export VARIABLE_NAME PATH=/bin:/usr/bin:. export PATH

Extracting values from variables To extract the value from variables, a dollar sign is used.

EXAMPLE

echo $variable_name echo $name echo $PATH

Rules : -

1.A variable name is any combination of alphabets, digits and an underscore („-„);

2.No commas or blanks are allowed within a variable name.

3.The first character of a variable name must either be an alphabet or an underscore.

4.Variables names should be of any reasonable length.

5.Variables name are case sensitive . That is , Name, NAME, name, NAme, are all different variables.

04. EXPRESSION Command :

To perform all arithematic operations .

Syntax : Var = „expr$value1‟ + $ value2‟

Arithmetic The Bourne shell does not support arithmetic. UNIX/Linux commands must be used to perform calculations.

EX NO:	
DATE:	

EXAMPLE

n=`expr 5 + 5` echo $n

Operators The Bourne shell uses the built-in test command operators to test numbers and strings.

EXAMPLE

Equality:

= *string* != *string* -eq *number* -ne *number*

Logical:

-a *and* -o *or* ! *not*

Logical:

AND &&

OR ||

Relational:

-gt *greater than* -ge *greater than, equal to*

-lt *less than* -le *less than, equal to*

Arithmetic **:**

+, -, *, /, %

Arguments (positional parameters) Arguments can be passed to a script from the command line. Positional parameters are used to receive their values from within the script.

EXAMPLE

At the command line:

$ scriptname arg1 arg2 arg3 ...

In a script:

echo $1 $2 $3 *Positional parameters* echo $* *All the positional paramters* echo $# *The number of positional parameters*

05.READ Statement :

To get the input from the user.

Syntax :

read x y

(no need of commas between variables)

06. ECHO Statement :

Similar to the output statement. To print output to the screen, the echo command is used. Wildcards must be escaped with either a backslash or matching quotes.

Syntax :

Echo “String” (or) echo $ b(for variable).

EXAMPLE

echo "What is your name?"

Reading user input The read command takes a line of input from the user and assigns it to a variable(s) on the right-hand side. The read command can accept muliple variable names. Each variable will be assigned a word.

EXAMPLE

echo "What is your name?" read name read name1 name2 ...

6. CONDITIONAL STATEMENTS :

The if construct is followed by a command. If an expression is to be tested, it is enclosed in square brackets. The then keyword is placed after the closing parenthesis. An if must end with a fi.

Syntax :

1.if This is used to check a condition and if it satisfies the condition if then does the next action , if not it goes to the else part.

2.if…else

Syntax :

If cp $ source $ target

EX NO:	
DATE:	

Then
Echo File copied successfully
Else
Echo Failed to copy the file.

3.nested if

here sequence of condition are checked and the corresponding performed accordingly.

Syntax :

```
if condition
then
command
if condition
then
command
else
command
fi
```

4.case ….. esac

This construct helps in execution of the shell script based on Choice.

EXAMPLE

The if construct is:

```
if command
then
block of statements
fi
```

```
if [ expression ]
then
block of statements
fi
```

The if/else/else if construct is:

```
if command
then
block of statements
elif command
then
block of statements
elif command
then
```

The case command construct is:

```
case variable_name in
pattern1)
statements
;;
pattern2)
statements
;;
pattern3)
;;
*) default value
;;
esac
case "$color" in
blue)
echo $color is blue
;;
green)
echo $color is green
;;
```

EX NO:	
DATE:	

block of statements
else
block of statements
fi
---
if [expression]
then
block of statements
elif [expression]
then
block of statements
elif [expression]
then
block of statements
else
block of statements
fi
---
do
block of statements
done
---
until control command
do
commands
done

red|orange)
echo $color is red or orange
;;
*) echo "Not a color" # default
esac

The if/else construct is:

if [expression]
then
block of statements
else
block of statements
fi
---

08. Break Statement :

This command is used to jump out of the loop instantly, without waiting to get the control command.

09. ARRAYS

(positional parameters) The Bourne shell does support an array, but a word list can be created by using positional parameters. A list of words follows the built-in set command, and the words are accessed by position. Up to nine positions are allowed.The built-in shift command shifts off the first word on the left-hand side of the list. The individual words are accessed by position values starting at 1.

EXAMPLE

set word1 word2 word3 echo $1 $2 $3 *Displays* word1, word2*, and* word3
set apples peaches plums shift *Shifts off* apples echo $1 *Displays first element of the list* echo $2 *Displays second element of the list* echo $* *Displays all elements of the list*

Command substitution To assign the output of a UNIX/Linux command to a variable, or use the output of a command in a string, backquotes are used.

EXAMPLE

variable_name=`command` echo $variable_name now=`date` echo $now echo "Today is `date`"

10. FILE TESTING

The Bourne shell uses the test command to evaluate conditional expressions and has a built-in set of options for testing attributes of files, such as whether it is a directory, a plain file (not a directory), a readable file, and so forth.

EXAMPLE

-d *File is a directory*
-f *File exists and is not a directory*

EX NO:	
DATE:	

–r *Current user can read the file*
–s *File is of nonzero size*
–w *Current user can write to the file*
–x *Current user can execute the file*

```
#!/bin/sh
1 if [ –f file ]
then
echo file exists
fi
2 if [ –d file ]
then
echo file is a directory
fi
3 if [ -s file ]
then
echo file is not of zero length
fi
4 if [ -r file -a -w file ]
then
echo file is readable and writable
fi
```

11. EXECUTION OF SHELL SCRIPT :

1.By using change mode command
2.$ chmod u + x sum.sh
3.$ sum.sh
or
$ sh sum.sh
or
$./sum.sh

RESULT:

EX NO:2.C

DATE:

SHELL PROGRAMMING
2.3.CONCATENATION OF TWO STRINGS

Aim:

Algorithm:

Program: CONCATENATION OF TWO STRINGS

```
echo “enter the first string”
read str1
echo “enter the second string”
read str2
echo “the concatenated string is” $str1$str2
```

EX NO:	
DATE:	

Input:
Enter first string: Hello
Enter first string: World

Output:
The concatenated string is HelloWorld

RESULT:

EX NO: 2.D

DATE:

SHELL PROGRAMMING
COMPARISON OF TWO STRINGS

Aim:

Algorithm:

PROGRAM:

```
echo “enter the first string”
read str1
echo “enter the second string”
read str2
if [ $str1 = $str2 ]
then
echo “strings are equal”
else
echo “strings are unequal”
fi
```

EX NO:	
DATE:	

INPUT:

Enter first string: hello

Enter second string: hello

OUTPUT:

The two strings are equal

INPUT:

Enter first string: hello

Enter second string: helo

OUTPUT:

The two strings are not equal

RESULT:

EX NO: 2.E

DATE:

SHELL PROGRAMMING
MAXIMUM OF THREE NUMBERS

Aim:

Algorithm:

PROGRAM:

```
echo "enter A"
read a
echo "enter B"
read b
echo "enter C"
read c
if [ $a -gt $b -a $a -gt $c ]
then
echo "A is greater"
elif [ $b -gt $a -a $b -gt $c ]
then
echo "B is greater"
else
echo "C is greater"
fi
```

EX NO:	
DATE:	

INPUT:

Enter A:23

Enter B:45

Enter C:67

OUTPUT:

C is greater

RESULT:

EX NO: 2.F

DATE:

SHELL PROGRAMMING
FIBONACCI SERIES

Aim:

Algorithm:

PROGRAM :

```
clear
  echo "How many number of terms to be generated ?"
  read n
  x=0
  y=1
  i=2
  echo "Fibonacci Series up to $n terms :"
  echo "$x"
  echo "$y"
  while [ $i -lt $n ]
  do
     i=`expr $i + 1 `
     z=`expr $x + $y `
     echo "$z"
     x=$y
     y=$z
  done
```

EX NO:	
DATE:	

INPUT:

How many number of terms to be generated?

5

OUTPUT:

Fibonacci Series up to 5 terms :

0

1

1

2

3

RESULT:

EX NO: 2.G

DATE:

ARITHMETIC OPERATIONS USING CASE

Aim:

Algorithm:

PROGRAM:

```
echo "Enter Two Numbers"
read a b
echo "What do you want to do? (1 to 5)"
echo "1) Sum"
echo "2) Difference"
echo "3) Product"
echo "4) Quotient"
echo "5) Remainder"
echo "Enter your Choice"
read n
case "$n" in
1) echo "The Sum of $a and $b is `expr $a + $b`";;
2) echo "The Difference between $a and $b is `expr $a - $b`";;
3) echo "The Product of the $a and $b is `expr $a \* $b`";;
4) echo "The Quotient of $a by $b is `expr $a / $b`";;
5) echo "The Remainder of $a by $b is `expr $a % $b`";;
esac
```

EX NO:	
DATE:	

INPUT1:
Enter Two Numbers
3 4
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice
1

OUTPUT1:
The Sum of 3 and 4 is 7

INPUT2:
Enter Two Numbers
3 4
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice
2

OUTPUT2:
The Difference between 3 and 4 is -1

RESULT :

EX NO: 2.H

DATE:

PROGRAM TO FIND ODD OR EVEN

Aim:

Algorithm:

PROGRAM:

```
echo -n "Enter number : "
read n
rem=$(( $n % 2 ))
if [ $rem -eq 0 ]
then
  echo "$n is even number"
else
  echo "$n is odd number"
fi
```

EX NO:	
DATE:	

OUTPUT:

Enter number : 3

3 is odd number

RESULT:

EX NO: 2.I

DATE:

PROGRAM TO FIND FACTORIAL OF A NUMBER

Aim:

Algorithm:

PROGRAM

```
fact ()
{
   if [ $1 -gt 1 ]; then
      y=`expr $1 - 1`
      fact $y
      x=$(( $1 * $? ))
      return $x
   else
      return 1
   fi
}

echo -e "Enter a number : \c"
read num
fact $num
echo "Factorial of $num is $?."
```

EX NO:	
DATE:	

OUTPUT:

Enter a number: 4

Factorial of 4 is 24.

RESULT:

EX NO: 3	Process Management using System Calls : Fork, Exit, Getpid, Wait, Close
DATE:	3.A.Program to simulate fork() System call

Aim:

Algorithm:

Program: fork() System call

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
  printf("I am: %d\n", (int) getpid());
  pid_t pid = fork();
  printf("fork returned: %d\n", (int) pid);

  if (pid < 0) { /* error occurred */
        perror("Fork failed");
  }
  if (pid == 0) { /* child process */
        printf("I am the child with pid %d\n", (int) getpid());
          printf("Child process is exiting\n");
          exit(0);
```

EX NO:	
DATE:	

```
    }
  /* parent process */
  printf("I am the parent waiting for the child process to end\n");
    wait(NULL);
    printf("parent process is exiting\n");
    return(0);
}
```

OUTPUT:

```
[cse2@localhost ~]$ ./a.out
I am: 3577
fork returned: 0
I am the child with pid 3578
Child process is exiting
fork returned: 3578
I am the parent waiting for the child process to end
parent process is exiting
```

RESULT:

EX NO:3.B	Program to simulate wait() System call
DATE:	

Aim:

Algorithm:

Program:

```
// C program to demonstrate working of wait()
#include<stdio.h>
#include<stdlib.h>
#include<sys/wait.h>
#include<unistd.h>

int main()
{
   pid_t cpid;
   if (fork()== 0)
      exit(0);          /* terminate child */
   else
      cpid = wait(NULL); /* reaping parent */
   printf("Parent pid = %d\n", getpid());
   printf("Child pid = %d\n", cpid);

   return 0;
}
```

EX NO:	
DATE:	

OUTPUT:

Parent pid = 12345678
Child pid = 89546848

RESULT:

EX NO:4	Implementation of the various CPU Scheduling Algorithms
DATE:	4.A.Implementation of FCFS Scheduling

Aim:

Algorithm:

PROGRAM-FCFS Scheduling

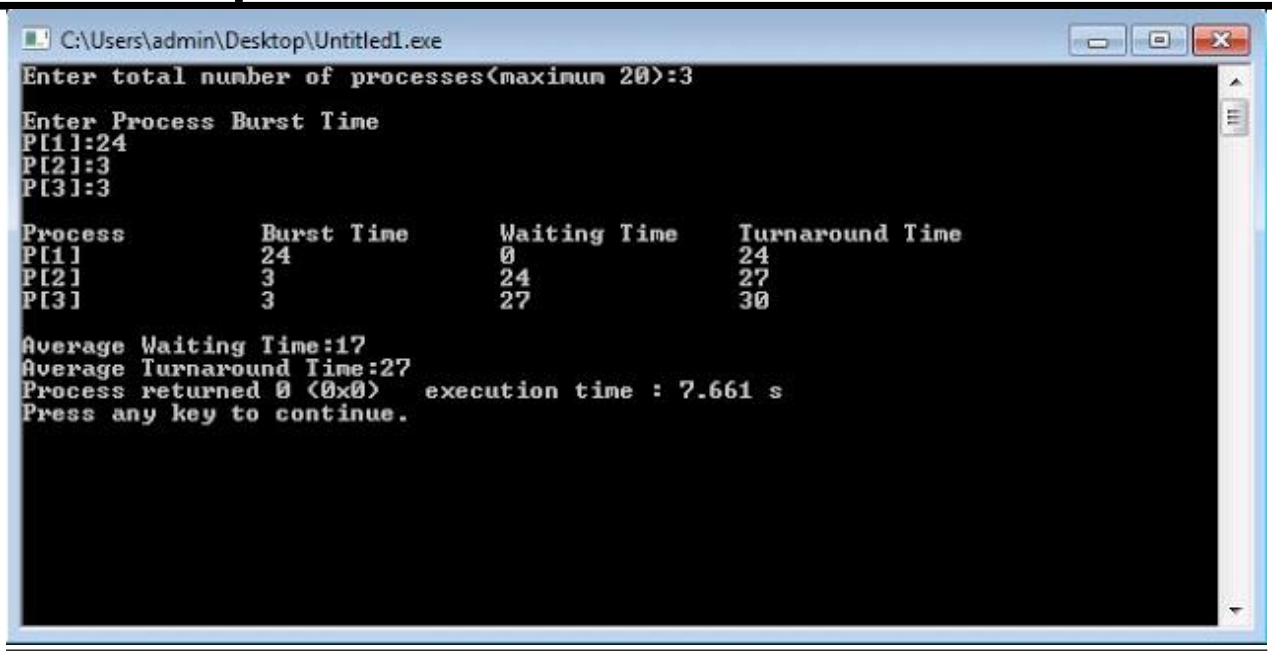
```
#include<stdio.h>
#include<string.h>
int main()
{
   int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
```

EX NO:	
DATE:	

```
    printf("Enter total number of processes(maximum 20):");
    scanf("%d",&n);

    printf("\nEnter Process Burst Time\n");
    for(i=0;i<n;i++)
    {
       printf("P[%d]:",i+1);
       scanf("%d",&bt[i]);
    }

    wt[0]=0;    //waiting time for first process is 0


    for(i=1;i<n;i++)
    {
       wt[i]=0;
       for(j=0;j<i;j++)
          wt[i]+=bt[j];
    }

    printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");


    for(i=0;i<n;i++)
    {
       tat[i]=bt[i]+wt[i];
       avwt+=wt[i];
       avtat+=tat[i];
       printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
    }
    avwt/=i;
    avtat/=i;
    printf("\n\nAverage Waiting Time:%d",avwt);
    printf("\nAverage Turnaround Time:%d",avtat);
     return 0;
}
```

OUTPUT:

EX NO:	
DATE:	



RESULT:

EX NO:4.B

DATE:

Implementation of SJF Scheduling

Aim:

Algorithm:

PROGRAM:-SJF Scheduling

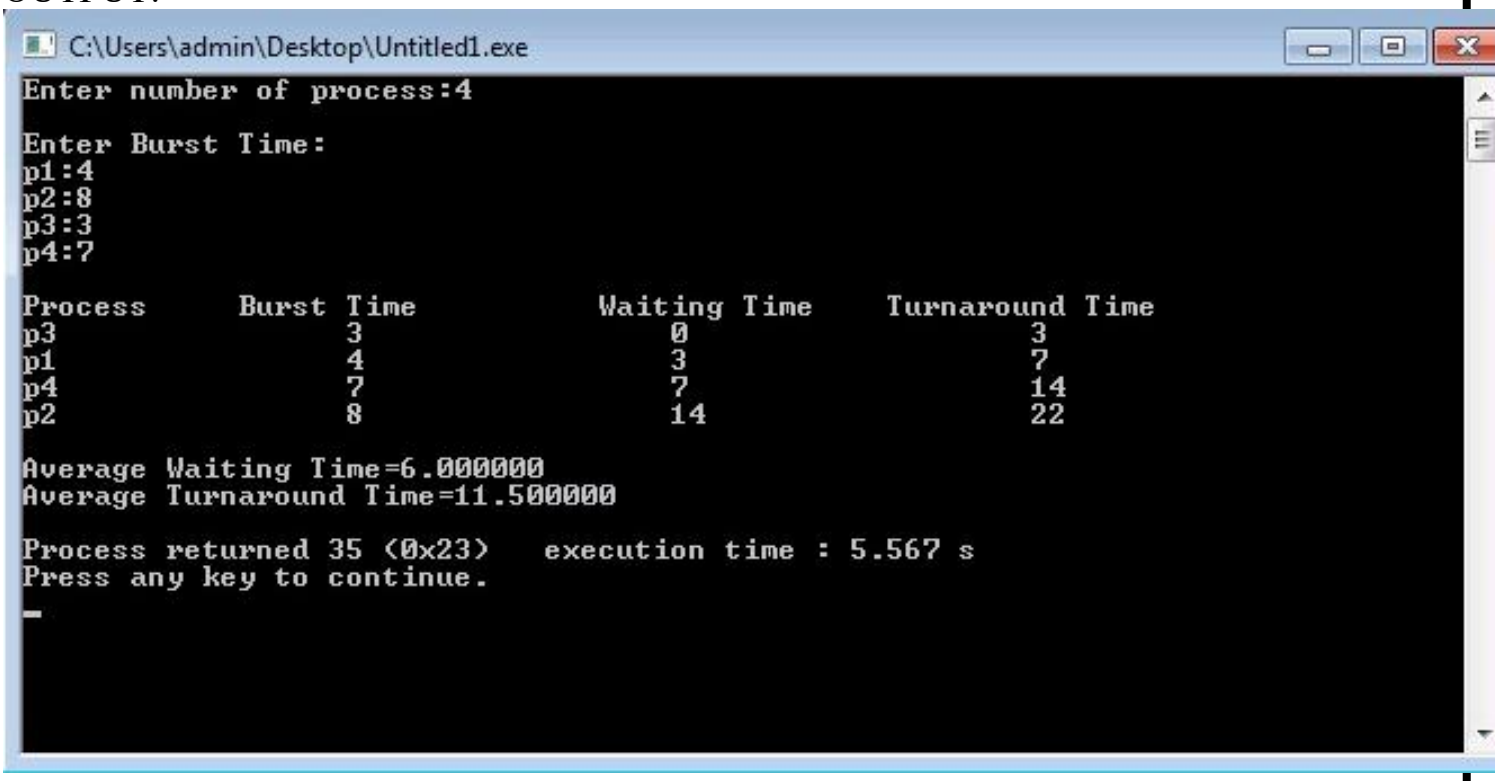
```
#include<stdio.h>
#include<string.h>
void main()
{
   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;
   printf("Enter number of process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time:\n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;          //contains process number
```

EX NO:	
DATE:	

```
    }

    //sorting burst time in ascending order using selection sort
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
        }

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;          //waiting time for first process will be zero

    //calculate waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=(float)total/n;      //average waiting time
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        total+=tat[i];
        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;     //average turnaround time
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

EX NO:	
DATE:	

OUTPUT:



RESULT:

EX NO: 4.C

DATE:

Implementation of Priority Scheduling

Aim:

Algorithm:

PROGRAM:-Priority Scheduling

```
#include<stdio.h>
#include<string.h>
 int main()
{
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   printf("Enter Total Number of Process:");
   scanf("%d",&n);

   printf("\nEnter Burst Time and Priority\n");
   for(i=0;i<n;i++)
   {
      printf("\nP[%d]\n",i+1);
```

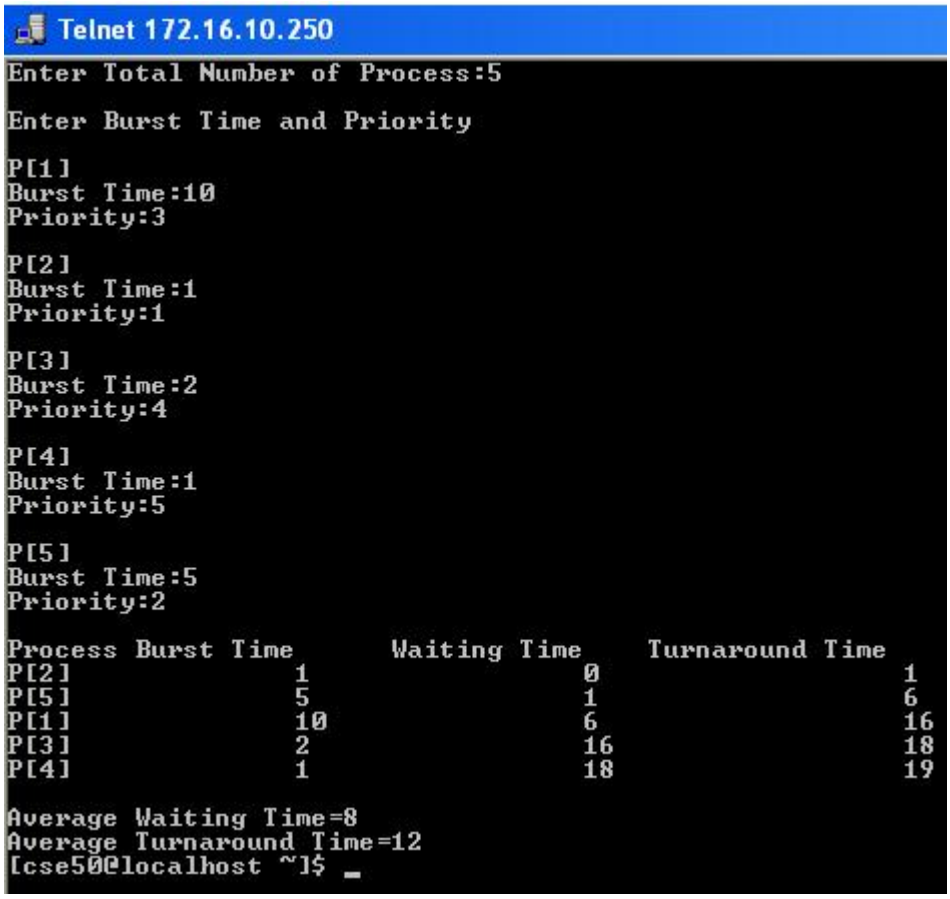
EX NO:	
DATE:	

```
    printf("Burst Time:");
    scanf("%d",&bt[i]);
    printf("Priority:");
    scanf("%d",&pr[i]);
    p[i]=i+1;           //contains process number
}

//sorting burst time, priority and process number in ascending order using selection sort
for(i=0;i<n;i++)
{
    pos=i;
    for(j=i+1;j<n;j++)
    {
        if(pr[j]<pr[pos])
            pos=j;
    }

    temp=pr[i];
    pr[i]=pr[pos];
    pr[pos]=temp;

    temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;

    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}

wt[0]=0;    //waiting time for first process is zero

//calculate waiting time
for(i=1;i<n;i++)
{
    wt[i]=0;
    for(j=0;j<i;j++)
        wt[i]+=bt[j];

    total+=wt[i];
}

avg_wt=total/n;      //average waiting time
total=0;

printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
    tat[i]=bt[i]+wt[i];     //calculate turnaround time
```

EX NO:	
DATE:	

```
        total+=tat[i];
        printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=total/n;     //average turnaround time
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);

    return 0;
}
```

OUTPUT:



RESULT:

EX NO: 4.D	Implementation of Round Robin Scheduling
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

Program: Round Robin

```
#include<stdio.h>
int main()
{
   int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
   float avg_wt, avg_tat;
   printf(" Total number of process in the system: ");
   scanf("%d", &NOP);
   y = NOP; // Assign the number of process to variable y

for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");
scanf("%d", &at[i]);
printf(" \nBurst time is: \t");
scanf("%d", &bt[i]);
temp[i] = bt[i];
}
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0)
{
   sum = sum + temp[i];
   temp[i] = 0;
   count=1;
   }
   else if(temp[i] > 0)
   {
      temp[i] = temp[i] - quant;
      sum = sum + quant;
   }
   if(temp[i]==0 && count==1)
   {
      y--;
```

EX NO:

DATE:

```
            printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
            wt = wt+sum-at[i]-bt[i];
            tat = tat+sum-at[i];
            count =0;
        }
        if(i==NOP-1)
        {
            i=0;
        }
        else if(at[i+1]<=sum)
        {
            i++;
        }
        else
        {
            i=0;
        }
    }
    avg_wt = wt * 1.0/NOP;
    avg_tat = tat * 1.0/NOP;
    printf("\n Average Waiting Time: \t%f", avg_wt);
    printf("\n Average Turnaround Time: \t%f", avg_tat);
    return 0;
    }
```

EX NO:	
DATE:	

OUTPUT:

```
 Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  8

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       1

Burst time is:  5

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       2

Burst time is:  10

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       3

Burst time is:  11
Enter the Time Quantum for the process:         6

 Process No              Burst Time              TAT             Waiting Time

 Process No[2]           5               10              5
 Process No[1]           8               25              17
 Process No[3]           10              27              17
 Process No[4]           11              31              20
 Average Turn Around Time:      14.750000
 Average Waiting Time:  23.250000
```

RESULT:

EX NO: 5	Inter process communication strategy
DATE:	5.A.Implementation of Shared Memory IPC

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:-Shared Memory IPC

SERVER:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/un.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define shmsize 27
int main()
{
char c;
int shmid;
key_t key = 2013;
char *shm, *s;
if ((shmid = shmget(key, shmsize, IPC_CREAT|0666)) < 0)
{
perror("shmget");
exit(1);
}
printf("Shared memory id : %d\n", shmid);
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)
{
perror("shmat");
exit(1);
}
memset(shm, 0, shmsize);
s = shm;
printf("Writing (a-z) onto shared memory\n");
```

EX NO:	
DATE:	

```
for (c = 'a'; c <= 'z'; c++)
*s++ = c;
*s = '\0';
while (*shm != '*');
printf("Client finished reading\n");
if(shmdt(shm) != 0)
fprintf(stderr, "Could not close memory segment.\n");
shmctl(shmid, IPC_RMID, 0);
return 0;
}
```

CLIENT:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define shmsize 27
int main()
{
int shmid;
key_t key = 2013;
char *shm, *s;
if ((shmid = shmget(key, shmsize, 0666)) < 0)
{
printf("Server not started\n");
exit(1);
}
else
printf("Accessing shared memory id : %d\n",shmid);
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)
{
perror("shmat");
exit(1);
}
printf("Shared memory contents:\n");
for (s = shm; *s != '\0'; s++)
putchar(*s);
putchar('\n');
*shm = '*';
return 0;
}
```

EX NO:	
DATE:	

OUTPUT:

SERVER:

```
[2cs254@cc5 ~]$ vi sserver.c
[2cs254@cc5 ~]$ cc sserver.c
[2cs254@cc5 ~]$ ./a.out
Shared memory id : 458756
Writing (a-z) onto shared memory
Client finished reading
```

CLIENT:

```
[2cs254@cc5 ~]$ vi client.c
[2cs254@cc5 ~]$ cc client.c
[2cs254@cc5 ~]$ ./a.out
shared memory id:458756
shared memory contents:
abcdefghijklmnopqurstuvwxyz
```

Result:

EX NO: 6	Implementation of mutual exclusion by Semaphore
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
int mutex=1,full=0,empty=3,x=0;
int main()
{
int n;
int producer();
int consumer();
int wait(int);
int signal(int);
printf("\n1.Producer \n2.Consumer \n3.Exit\n");
while(1)
{
printf("\n Enter your choice\n");
scanf("%d",&n);
switch(n)
{
case 1:
if((mutex==1)&&(empty!=0))
producer();
else
printf("\n Buffer is Full\n");
break;
case 2:
if((mutex==1)&&(full!=0))
consumer();
else
printf("\n Buffer is Empty\n");
break;
case 3:
exit(0);
break;
}
}
}
int wait(int s)
{
return(--s);
}
int signal(int s)
{
return(++s);
}
producer()
{
mutex=wait(mutex);
full=signal(full);
```

EX NO:	
DATE:	

```
empty=wait(empty);
x++;
printf("\n Producer Produces the item%d\n",x);
mutex=signal(mutex);
}
consumer()
{
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\n Consumer consumes item%d\n",x);
x--;
mutex=signal(mutex);
}
```

OUTPUT:

```
[2cs163@cc5]$ vi semaphore.c
[2cs163@cc5]$ cc semaphore.c
 [2cs163@cc5]$ ./a.out
1.Producer
2.Consumer
3.Exit
 Enter your choice
1
 Producer Produces the item1
 Enter your choice
2
 Consumer consumes item1
 Enter your choice
3
[2cs163@cc5 ~]$ ./a.out
1.Producer
2.Consumer
3.Exit
 Enter your choice
2

 Buffer is Empty
 Enter your choice
3
```

RESULT:

EX NO: 7	Bankers algorithm for deadlock avoidance
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:-Bankers Deadlock Avoidance

```
// Banker's Algorithm
#include <stdio.h>
int main()
{
   // P0, P1, P2, P3, P4 are the Process names here

   int n, m, i, j, k;
   n = 5; // Number of processes
   m = 3; // Number of resources
   int alloc[5][3] = { { 0, 1, 0 }, // P0    // Allocation Matrix
                       { 2, 0, 0 }, // P1
                       { 3, 0, 2 }, // P2
                       { 2, 1, 1 }, // P3
                       { 0, 0, 2 } }; // P4

   int max[5][3] = { { 7, 5, 3 }, // P0    // MAX Matrix
                     { 3, 2, 2 }, // P1
                     { 9, 0, 2 }, // P2
                     { 2, 2, 2 }, // P3
                     { 4, 3, 3 } }; // P4

   int avail[3] = { 3, 3, 2 }; // Available Resources

   int f[n], ans[n], ind = 0;
   for (k = 0; k < n; k++) {
      f[k] = 0;
   }
   int need[n][m];
   for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++)
         need[i][j] = max[i][j] - alloc[i][j];
   }
   int y = 0;
   for (k = 0; k < 5; k++) {
      for (i = 0; i < n; i++) {
         if (f[i] == 0) {

            int flag = 0;
            for (j = 0; j < m; j++) {
```

EX NO:	
DATE:	

```
                if (need[i][j] > avail[j]){
                    flag = 1;
                     break;
                }
            }

            if (flag == 0) {
                ans[ind++] = i;
                for (y = 0; y < m; y++)
                    avail[y] += alloc[i][y];
                f[i] = 1;
            }
        }
    }
}

  int flag = 1;

  for(int i=0;i<n;i++)
{
  if(f[i]==0)
  {
    flag=0;
     printf("The following system is not safe");
    break;
  }
}

  if(flag==1)
{
  printf("Following is the SAFE Sequence\n");
  for (i = 0; i < n - 1; i++)
    printf(" P%d ->", ans[i]);
  printf(" P%d", ans[n - 1]);
}


  return (0);


}
```

EX NO:	
DATE:	

OUT PUT:

P1 -> P3 -> P4 -> P0 -> P2

RESULT:

EX NO:8	Implementation of Deadlock Detection Algorithm
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:Deadlock Detection

```
#include <stdio.h>
#include <stdbool.h>
#define N 5
#define M 3

int main()
{
   int allocation[N][M] = {{0, 1, 0},
                           {2, 0, 0},
                           {0, 0, 2},
                           {1, 1, 0},
                           {0, 0, 2}};

   int max[N][M] = {{1, 1, 1},
                    {2, 0, 0},
                    {0, 0, 2},
                    {1, 2, 0},
                    {2, 2, 2}};

   int available[M] = {0, 0, 0};

   int need[N][M];
   for (int i = 0; i < N; i++) {
      for (int j = 0; j < M; j++) {
         need[i][j] = max[i][j] - allocation[i][j];
      }
   }

   bool finished[N] = {false};
   bool deadlock = true;

   while (deadlock) {
      deadlock = false;

      for (int i = 0; i < N; i++) {
         if (!finished[i]) {
            bool can_run = true;

            for (int j = 0; j < M; j++) {
               if (need[i][j] > available[j]) {
```

EX NO:	
DATE:	

```
                    can_run = false;
                    break;
                }
            }

            if (can_run) {
                finished[i] = true;

                for (int j = 0; j < M; j++) {
                    available[j] += allocation[i][j];
                }
            }
        }
    }

    for (int i = 0; i < N; i++) {
        if (!finished[i]) {
            bool can_run = true;

            for (int j = 0; j < M; j++) {
                if (need[i][j] > available[j]) {
                    can_run = false;
                    break;
                }
            }

            if (can_run) {
                deadlock = true;
                printf("Deadlock detected in process P%d\n", i);
                break;
            }
        }
    }
}

    return 0;
}
```

EX NO:	
DATE:	

OUTPUT:
Deadlock detected in process P0

RESULT:

EX NO: 9	Implementation of Threading & Synchronization Applications
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

Program: Threading & Synchronization Applications

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
pthread_t tid[2];
int counter;
pthread_mutex_t lock;
void * doSomeThing(void *arg)
{
    pthread_mutex_lock(&lock);
    unsigned long i = 0;
    counter += 1;
    printf("\n Job %d started\n", counter);

    for(i=0; i<(0xFFFFFFFF);i++);

    printf("\n Job %d finished\n", counter);

    pthread_mutex_unlock(&lock);

    return NULL;
}
int main()
{
    int i = 0;
    int err;

    if (pthread_mutex_init(&lock, NULL) != 0)
    {
        printf("\n mutex init failed\n");
        return 1;
    }

    while(i < 2)
```

EX NO:	
DATE:	

```
    {
        err = pthread_create(&(tid[i]), NULL, &doSomeThing, NULL);
        if (err != 0)
            printf("\ncan't create thread :[%s]", strerror(err));
        i++;
    }
    pthread_join(tid[0], NULL);
    pthread_join(tid[1], NULL);
    pthread_mutex_destroy(&lock);
    return 0;
}
```

To Compile this Program,give the following command:
$] gcc filename.c –lpthread
To run that above program
$]./a.out

Output:
$ ./threads

Job 1 started
Job 1 finished
Job 2 started
Job 2 finished

Explanation:

The loop `for(i=0; i<(0xFFFFFFFF);i++);` in the `doSomeThing` function is used to simulate a long-running task that takes some time to complete. The loop simply iterates a very large number of times (0xFFFFFFFF, which is the hexadecimal representation of the decimal number 4,294,967,295) without actually doing anything in each iteration. This is done to introduce a delay between the "started" and "finished" messages that are printed by the thread.

The `pthread_mutex_init` function is used to initialize a mutex object before it can be used for synchronizing access to shared resources in a multi-threaded program.

EX NO:	
DATE:	

In the code snippet you provided, the `pthread_mutex_init` function is used to initialize the `lock` mutex object with default attributes. If the initialization fails (indicated by the function returning a non-zero value), the program prints an error message and returns with an exit status of 1.

It's important to initialize a mutex object before using it, as uninitialized mutexes can lead to undefined behavior and potential race conditions in a multi-threaded program. The `pthread_mutex_init` function ensures that the mutex is properly initialized and ready to use.

Result:

EX NO:10	Implementation of Threading & Synchronization Applications
DATE:	10.A.FIFO Page Replacement

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
        printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
        printf("\n ENTER THE PAGE NUMBER :\n");
        for(i=1;i<=n;i++)
        scanf("%d",&a[i]);
        printf("\n ENTER THE NUMBER OF FRAMES :");
        scanf("%d",&no);
for(i=0;i<no;i++)
        frame[i]= -1;
                j=0;
                printf("\tref string\t page frames\n");
for(i=1;i<=n;i++)
                {
                        printf("%d\t\t",a[i]);
                        avail=0;
                        for(k=0;k<no;k++)
if(frame[k]==a[i])
                                avail=1;
                        if (avail==0)
                        {
                                frame[j]=a[i];
                                j=(j+1)%no;
                                count++;
                                for(k=0;k<no;k++)
                                printf("%d\t",frame[k]);
}
                        printf("\n");
}
                printf("\nPage Fault Is %d\n",count);
                return 0;
}
```

EX NO:	
DATE:	

OUTPUT:

ENTER THE NUMBER OF PAGES: 20

ENTER THE PAGE NUMBER : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

ENTER THE NUMBER OF FRAMES :3

ref string page frames

7	7	-1	-1
0	7	0	-1
1	7	0	1
2	2	0	1
0			
3	2	3	1
0	2	3	0
4	4	3	0
2	4	2	0
3	4	2	3
0	0	2	3
3			
2			
1	0	1	3
2	0	1	2
0			
1			
7	7	1	2
0	7	0	2
1	7	0	1

Page Fault Is 15

RESULT:

EX NO: 10.B	LRU Page replacement
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:LRU Page replacement

```
#include<stdio.h>
Int main()
{
int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
printf("Enter no of pages:");
scanf("%d",&n);
printf("Enter the reference string:");
for(i=0;i<n;i++)
        scanf("%d",&p[i]);
printf("Enter no of frames:");
scanf("%d",&f);
q[k]=p[k];
printf("\n\t%d\n",q[k]);
c++;
k++;
for(i=1;i<n;i++)
        {
                c1=0;
                for(j=0;j<f;j++)
                {
                        if(p[i]!=q[j])
                        c1++;
                }
                if(c1==f)
                {
                        c++;
                        if(k<f)
                        {
                                q[k]=p[i];
                                k++;
                                for(j=0;j<k;j++)
                                printf("\t%d",q[j]);
                                printf("\n");
                        }
                        else
                        {
                                for(r=0;r<f;r++)
                                {
                                        c2[r]=0;
                                        for(j=i-1;j<n;j--)
                                        {
                                        if(q[r]!=p[j])
                                        c2[r]++;
                                        else
                                        break;
                                }
```

EX NO:

DATE:

```
                        }
                        for(r=0;r<f;r++)
                         b[r]=c2[r];
                        for(r=0;r<f;r++)
                        {
                                for(j=r;j<f;j++)
                                {
                                        if(b[r]<b[j])
                                        {
                                                t=b[r];
                                                b[r]=b[j];
                                                b[j]=t;
                                        }
                                }
                        }
                        for(r=0;r<f;r++)
                        {
                                if(c2[r]==b[0])
                                q[r]=p[i];
                                printf("\t%d",q[r]);
                        }
                        printf("\n");
                }
        }
}
printf("\nThe no of page faults is %d",c);
}
```

EX NO:	
DATE:	

OUTPUT:

Enter no of pages:10

Enter the reference string:7 5 9 4 3 7 9 6 2 1

Enter no of frames:3

7		
7	5	
7	5	9
4	5	9
4	3	9
4	3	7
9	3	7
9	6	7
9	6	2
1	6	2

The no of page faults is 10

RESULT:

EX NO: 10.C	LFU Page replacement
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
 int main()
{
    int total_frames, total_pages, hit = 0;
    int pages[25], frame[10], arr[25], time[25];
    int m, n, page, flag, k, minimum_time, temp;
    printf("Enter Total Number of Pages:\t");
    scanf("%d", &total_pages);
    printf("Enter Total Number of Frames:\t");
    scanf("%d", &total_frames);
    for(m = 0; m < total_frames; m++)
    {
        frame[m] = -1;
    }
    for(m = 0; m < 25; m++)
    {
        arr[m] = 0;
    }
    printf("Enter Values of Reference String\n");
    for(m = 0; m < total_pages; m++)
    {
        printf("Enter Value No.[%d]:\t", m + 1);
        scanf("%d", &pages[m]);
    }
    printf("\n");
    for(m = 0; m < total_pages; m++)
    {
        arr[pages[m]]++;
        time[pages[m]] = m;
        flag = 1;
        k = frame[0];
        for(n = 0; n < total_frames; n++)
        {
            if(frame[n] == -1 || frame[n] == pages[m])
            {
                if(frame[n] != -1)
                {
                    hit++;
                }
```

EX NO:	
DATE:	

```
                flag = 0;
                frame[n] = pages[m];
                break;
            }
            if(arr[k] > arr[frame[n]])
            {
                k = frame[n];
            }
        }
        if(flag)
        {
            minimum_time = 25;
            for(n = 0; n < total_frames; n++)
            {
                if(arr[frame[n]] == arr[k] && time[frame[n]] < minimum_time)
                {
                    temp = n;
                    minimum_time = time[frame[n]];
                }
            }
            arr[frame[temp]] = 0;
            frame[temp] = pages[m];
        }
        for(n = 0; n < total_frames; n++)
        {
            printf("%d\t", frame[n]);
        }
        printf("\n");
    }
    printf("Page Hit:\t%d\n", hit);
}
```

EX NO:	
DATE:	

OUTPUT:

```
Enter Total Number of Frames:   3
Enter Values of Reference String
Enter Value No.[1]:     2
Enter Value No.[2]:     3
Enter Value No.[3]:     4
Enter Value No.[4]:     2
Enter Value No.[5]:     1
Enter Value No.[6]:     3
Enter Value No.[7]:     7
Enter Value No.[8]:     5
Enter Value No.[9]:     4
Enter Value No.[10]:    3

2       -1      -1
2       3       -1
2       3       4
2       3       4
2       1       4
2       1       3
2       7       3
2       7       5
2       4       5
2       4       3
Page Hit:       1
```

RESULT:

EX NO: 10.D	Optimal Page replacement
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

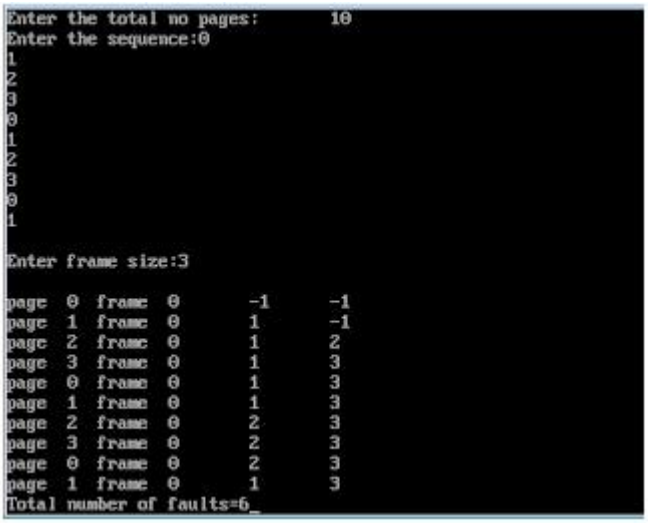
```
#include<stdio.h>
int main()
{
int n,pg[30],fr[10];
int count[10],i,j,k,fault,f,flag,temp,current,c,dist,max,m,cnt,p,x;
fault=0;
dist=0;
k=0;
printf("Enter the total no pages:\t");
scanf("%d",&n);
printf("Enter the sequence:");
for(i=0;i<n;i++)
scanf("%d",&pg[i]);
printf("\nEnter frame size:");
scanf("%d",&f);

for(i=0;i<f;i++)
{
count[i]=0;
fr[i]=-1;
}
for(i=0;i<n;i++)
{
flag=0;
temp=pg[i];
for(j=0;j<f;j++)
{
if(temp==fr[j])
{
flag=1;
break;
}
}
if((flag==0)&&(k<f))
{
fault++;
fr[k]=temp;
k++;
}
else if((flag==0)&&(k==f))
{
fault++;
for(cnt=0;cnt<f;cnt++)
{
current=fr[cnt];
for(c=i;c<n;c++)
{
```

EX NO:	
DATE:	

```
if(current!=pg[c])
count[cnt]++;
else
break;
}
}
max=0;
for(m=0;m<f;m++)
{
if(count[m]>max)
{
max=count[m];
p=m;
}
}
fr[p]=temp;
}
printf("\npage  %d  frame\t",pg[i]);
for(x=0;x<f;x++)
{
printf("%d\t",fr[x]);
}
}
printf("\nTotal number of faults=%d",fault);
return 0;
}
```

EX NO:	
DATE:	

OUTPUT



RESULT:

EX NO: 11	Write C programs to implement the following Memory Allocation Methods
DATE:	11.A.First Fit

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
void firstFit(int blockSize[], int m, int processSize[], int n)
{
    int i, j;
    int allocation[n];
    for(i = 0; i < n; i++)
    {
        allocation[i] = -1;
    }

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++)
    {
        printf(" %i\t\t\t", i+1);
        printf("%i\t\t\t\t", processSize[i]);
        if (allocation[i] != -1)
            printf("%i", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
```

EX NO:	
DATE:	

```
}


int main()
{
   int m;
     int n;
     int blockSize[] = {100, 500, 200, 300, 600};
   int processSize[] = {212, 417, 112, 426};
   m = sizeof(blockSize) / sizeof(blockSize[0]);
   n = sizeof(processSize) / sizeof(processSize[0]);
   firstFit(blockSize, m, processSize, n);
   return 0 ;
}
```

Input and Output:

Input : blockSize[] = {100, 500, 200, 300, 600};
processSize[] = {212, 417, 112, 426};
Output:

Process No.	Process Size	Block no.
1	212	2
2	417	5
3	112	3
4	426	Not Allocated

OUTPUT:

Process No.	Process Size	Block no.
1	212	2
2	417	5
3	112	2
4	426	Not Allocated

RESULT:

EX NO: 11.B	Worst Fit
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include <stdio.h>
#include <string.h>
void worstFit(int blockSize[], int m, int processSize[], int n)
{

    int allocation[n];
    memset(allocation, -1, sizeof(allocation));
    for (int i = 0; i < n; i++)
    {
        int wstIdx = -1;
        for (int j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (wstIdx == -1)
                    wstIdx = j;
                else if (blockSize[wstIdx] < blockSize[j])
                    wstIdx = j;
            }
        }

        if (wstIdx != -1)
        {
            allocation[i] = wstIdx;
            blockSize[wstIdx] -= processSize[i];
        }
    }
    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++)
    {
        printf("   %d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}

int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);
```

EX NO:	
DATE:	

```
    worstFit(blockSize, m, processSize, n);
    return 0;
}
```

OUTPUT:

```
Process No.     Process Size    Block no.
   1            212             5
   2            417             2
   3            112             5
   4            426             Not Allocated
```

RESULT:

EX NO: 11.C	Best Fit
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>

// Method to allocate memory to blocks as per Best fit algorithm
void bestFit(int blockSize[], int m, int processSize[], int n)
{
    // Stores block id of the block allocated to a process
    int allocation[n];

    // Initially no block is assigned to any process
    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i = 0; i < n; i++)
    {
        // Find the best fit block for current process
        int bestIdx = -1;
        for (int j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (bestIdx == -1)
                    bestIdx = j;
                else if (blockSize[bestIdx] > blockSize[j])
                    bestIdx = j;
            }
        }

        // If we could find a block for current process
        if (bestIdx != -1)
        {
            // allocate block j to p[i] process
            allocation[i] = bestIdx;

            // Reduce available memory in this block.
            blockSize[bestIdx] -= processSize[i];
        }
    }
```

EX NO:	
DATE:	

```
    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++)
    {
        printf(" %d\t\t%d\t\t", i+1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}

// Driver Method
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    bestFit(blockSize, m, processSize, n);

    return 0 ;
}
```

EX NO:	
DATE:	

INPUT & OUTPUT:

Input : blockSize[] = {100, 500, 200, 300, 600};
processSize[] = {212, 417, 112, 426};

Output:

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5

OUTPUT:

Process No.	Process Size	Block no.
1	212	4
2	417	2
3	112	3
4	426	5

RESULT:

EX NO: 12	Implementation of the paging Technique
DATE:	

Aim:

Algorithm:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
int main()
{
 int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;
 int s[10], fno[10][20];
printf("\nEnter the memory size -- ");
scanf("%d",&ms);
printf("\nEnter the page size -- ");
scanf("%d",&ps);
nop = ms/ps;
printf("\nThe no. of pages available in memory are -- %d ",nop);
printf("\nEnter number of processes -- ");
 scanf("%d",&np);
rempages = nop;
for(i=1;i<=np;i++)
{
printf("\nEnter no. of pages required for p[%d]-- ",i);
 scanf("%d",&s[i]);

if(s[i] >rempages)
{
printf("\nMemory is Full");
break;
}
rempages = rempages - s[i];
printf("\nEnter pagetable for p[%d] --- ",i);
 for(j=0;j<s[i];j++)
scanf("%d",&fno[i][j]);
}
printf("\nEnter Logical Address to find Physical Address ");
printf("\nEnter process no. and pagenumber and offset -- ");
scanf("%d %d %d",&x,&y, &offset);
if(x>np || y>=s[i] || offset>=ps)
printf("\nInvalid Process or Page Number or offset");
else
{ pa=fno[x][y]*ps+offset;
printf("\nThe Physical Address is -- %d",pa);
}
}
```

EX NO:	
DATE:	

INPUT:

Enter the memory size – 1000 Enter the page size -- 100
The no. of pages available in memory are -- 10
Enter number of processes -- 3
Enter no. of pages required for p[1]-- 4
Enter pagetable for p[1] --- 8 6
9
5
Enter no. of pages required for p[2]-- 5
Enter pagetable for p[2] --- 1 4 5 7 3
Enter no. of pages required for p[3]-- 5

OUTPUT

Memory is Full
Enter Logical Address to find Physical Address Enter process no. and pagenumber and offset –
2
3
60
The Physical Address is -- 760

RESULT:

EX NO: 13	WRITE C PROGRAMS TO IMPLEMENT THE VARIOUS FILE ORGANIZATION TECHNIQUES
DATE:	13.A) C Program to Implement Sequential File Allocation Technique

Aim:

Algorithm:

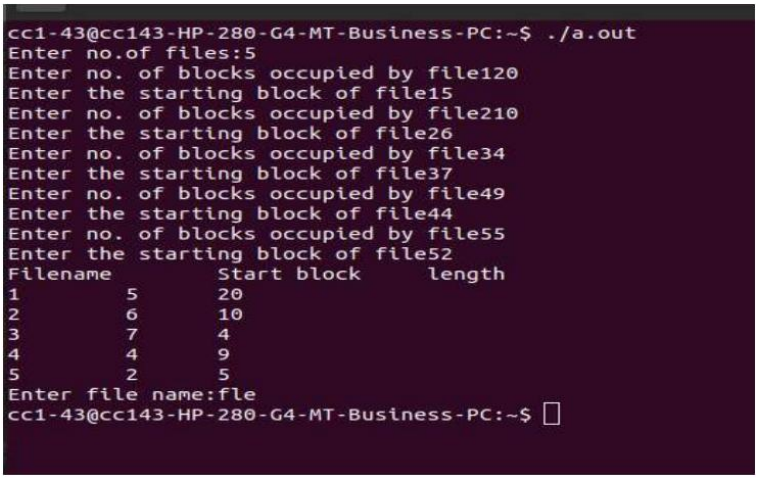
EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
int main()
{
int n,i,j,b[20],sb[20],t[20],x,c[20][20];
printf("Enter no.of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter no. of blocks occupied by file%d",i+1);
scanf("%d",&b[i]);
printf("Enter the starting block of file%d",i+1);
scanf("%d",&sb[i]);
t[i]=sb[i];
for(j=0;j<b[i];j++)
c[i][j]=sb[i]++;
}
printf("Filename\tStart block\tlength\n");
for(i=0;i<n;i++)
printf("%d\t %d \t%d\n",i+1,t[i],b[i]);
printf("Enter file name:");
scanf("%d",&x);
printf("File name is:%d",x);
printf("length is:%d",b[x-1]);
printf("blocks occupied:");
for(i=0;i<b[x-1];i++)
printf("%4d",c[x-1][i]);
}
```

EX NO:	
DATE:	

OUTPUT:



RESULT:

EX NO: 13.B

DATE:

C Program to Implement Indexed File Allocation Technique

AIM:

ALGORITHM:

PROGRAM:

```
#include<stdio.h>
int main()
{
int n,m[20],i,j,sb[20],s[20],b[20][20],x;
printf("Enter no. of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter starting block and size of file%d:",i+1);
```

EX NO:	
DATE:	

```
scanf("%d%d",&sb[i],&s[i]);
printf("Enter blocks occupied by file%d:",i+1);
scanf("%d",&m[i]);
printf("enter blocks of file%d:",i+1);
for(j=0;j<m[i];j++)
scanf("%d",&b[i][j]);
}
printf("\nFile\t index\tlength\n");
for(i=0;i<n;i++)
{
printf("%d\t%d\t%d\n",i+1,sb[i],m[i]);
}
printf("\nEnter file name:");
scanf("%d",&x);
printf("file name is:%d\n",x);
i=x-1;
printf("Index is:%d",sb[i]);
printf("Block occupiedare:");
for(j=0;j<m[i];j++)
printf("%d",b[i][j]);
}
```

OUTPUT:

```
cc1-43@cc143-HP-280-G4-MT-Business-PC:~$ ./a.out
Enter no. of files:2
Enter starting block and size of file1:2 5
Enter blocks occupied by file1:10
enter blocks of file1:3
2 5 4 6 7 2 6 4 7
Enter starting block and size of file2:3 4
Enter blocks occupied by file2:5
enter blocks of file2:2 3 4 5 6

File     index  length
1       2       10
2       3       5

Enter file name:1
file name is:1
cc1-43@cc143-HP-280-G4-MT-Business-PC:~$
```

RESULT:

EX NO:13.C	C Program to Implement Linked File Allocation Technique
DATE:	

AIM:

ALGORITHM:

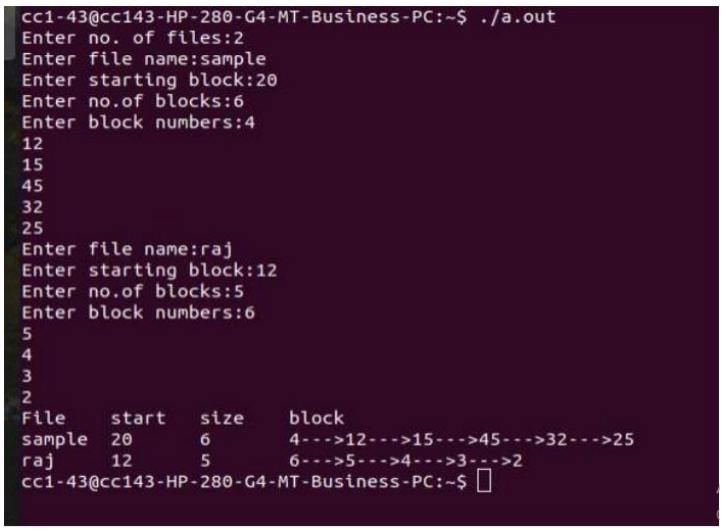
EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
int main()
{
int n,m[20],i,j,sb[20],s[20],b[20][20],x;
printf("Enter no. of files:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter starting block and size of file%d:",i+1);
scanf("%d%d",&sb[i],&s[i]);
printf("Enter blocks occupied by file%d:",i+1);
scanf("%d",&m[i]);
printf("enter blocks of file%d:",i+1);
for(j=0;j<m[i];j++)
scanf("%d",&b[i][j]);
}
printf("\nFile\t index\tlength\n");
for(i=0;i<n;i++)
{
printf("%d\t%d\t%d\n",i+1,sb[i],m[i]);
}
printf("\nEnter file name:");
scanf("%d",&x);
printf("file name is:%d\n",x);
i=x-1;
printf("Index is:%d",sb[i]);
printf("Block occupiedare:");
for(j=0;j<m[i];j++)
printf("%d",b[i][j]);
}
```

EX NO:	
DATE:	

OUTPUT:



RESULT:

EX NO: 14	Implementation of the following File Allocation Strategies
DATE:	14.A . SEQUENTIAL FILE ALLOCATION

AIM:

ALGORITHM:

EX NO:	
DATE:	

PROGRAM: SEQUENTIAL FILE ALLOCATION

```
#include<stdio.h>
int main()
{
        int nf, fc[20], mb[100], i, j, k, fb[100], fs[20], mc=0;
        printf("\nEnter the number of files:"); scanf("%d",&nf);
        for(i=0;i<nf;i++)
        {
                printf("\nEnter the capacity of file %d: ",i+1); scanf("%d",&fc[i]);
                printf("\nEnter the starting address of file %d: ",i+1); scanf("%d",&fs[i]);
        }
        printf("\n---SEQUENTIAL  FILE ALLOCATION---\n"); for(i=0;i<100;i++)
        fb[i]=1;
        for(i=0;i<nf;i++)
        {
                j=fs[i];

                        {
                        if(fb[j]==1)
                        {
                                for(k=j;k<(j+fc[i]);k++)
                                {
                                        if(fb[k]==1)
                                        mc++;
                                }
                                if(mc==fc[i])
                                {
                                        for(k=fs[i];k<(fs[i]+fc[i]);k++)
                                        {
                                                fb[k]=0;
                                        }
                                        printf("\nFile %d allocated in memory %d
                                to %d...",i+1,fs[i],fs[i]+fc[i]-1);
                                }
                        }
                        else
                        printf("\nFile %d not allocated since %d contiguous memory not available
                        from %d...",i+1,fc[i],fs[i]);
        }
        mc=0;
        return 0;
}}
```

EX NO:	
DATE:	

OUTPUT:

Enter the number of files: 3 Enter the capacity
of file 1: 21
Enter the starting address of file 1: 21 Enter the
capacity of file 2: 24
Enter the starting address of file 2: 36 Enter the
capacity of file 3: 2
Enter the starting address of file 3: 54
---SEQUENTIAL FILE ALLOCATION---
File 1 allocated in memory 21 to 41...
File 2 not allocated since 24 contiguous memory not available from 36...
File 3 allocated in memory 54 to 55...

RESULT:

EX NO: 14 .B

DATE:

INDEXED FILE ALLOCATION

AIM:

ALGORITHM:

EX NO:	
DATE:	

Program Code:Indexed File Allocation

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
x:printf("Enter the index block: ");
scanf("%d",&ind);
if(f[ind]!=1)
{
printf("Enter no of blocks needed and no of files for the index %d on the disk : \n", ind);
scanf("%d",&n);
}
else
{
printf("%d index is already allocated \n",ind);
goto x;
}
y: count=0;
for(i=0;i<n;i++)
{
scanf("%d", &index[i]);
if(f[index[i]]==0)
count++;
}
if(count==n)
{
for(j=0;j<n;j++)
f[index[j]]=1;
printf("Allocated\n");
printf("File Indexed\n");
for(k=0;k<n;k++)
printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);
}
else
{
printf("File in the index is already allocated \n");
printf("Enter another file indexed");
goto y;
}
```

EX NO:	
DATE:	

```
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit(0);
getch();
}
```

Program Output:

```
Enter the index block: 5
Enter no of blocks needed and no of files for the index 5 on the disk :
4
1 2 3 4
Allocated
File Indexed
5-------->1 : 1
5-------->2 : 1
5-------->3 : 1
5-------->4 : 1
Do you want to enter more file(Yes - 1/No - 0)1
Enter the index block: 4
4 index is already allocated
Enter the index block: 6
Enter no of blocks needed and no of files for the index 6 on the disk :
2
7 8
A5llocated
File Indexed
6-------->7 : 1
6-------->8 : 1
Do you want to enter more file(Yes - 1/No - 0)0
```

RESULT:

EX NO: 14.C	Linked File Allocation
DATE:	

AIM:

ALGORITHM:

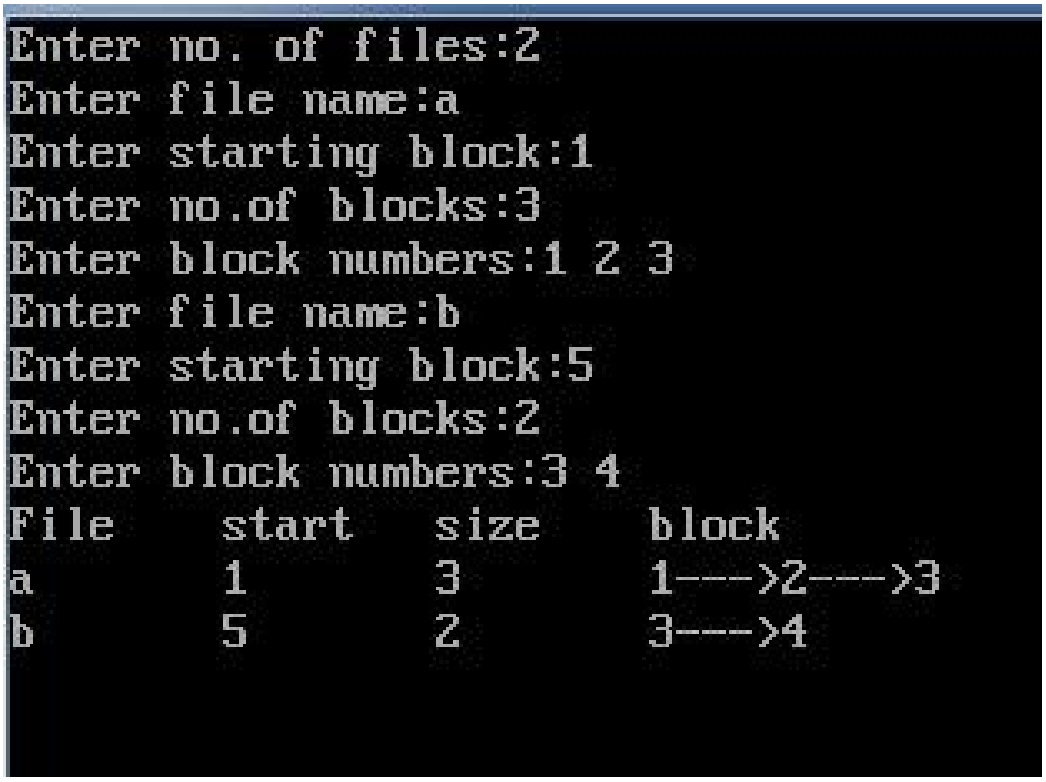
EX NO:	
DATE:	

PROGRAM: **Linked File Allocation**

```
#include<stdio.h>
struct file
{
 char fname[10];
 int start,size,block[10];
}f[10];
int main()
{
 int i,j,n;
 printf("Enter no. of files:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
 printf("Enter file name:");
 scanf("%s",&f[i].fname);
 printf("Enter starting block:");
 scanf("%d",&f[i].start);
 f[i].block[0]=f[i].start;
 printf("Enter no.of blocks:");
 scanf("%d",&f[i].size);
 printf("Enter block numbers:");
 for(j=1;j<=f[i].size;j++)
 {
        scanf("%d",&f[i].block[j]);
 }
 }
 printf("File\tstart\tsize\tblock\n");
 for(i=0;i<n;i++)
 {
        printf("%s\t%d\t%d\t",f[i].fname,f[i].start,f[i].size);
        for(j=1;j<=f[i].size-1;j++)
                printf("%d--->",f[i].block[j]);
        printf("%d",f[i].block[j]);
        printf("\n");
 }
return 0;
}
```

EX NO:	
DATE:	

OUTPUT:



RESULT:

EX NO: 15	Write C programs for the implementation of various disk scheduling algorithms
DATE:	15.A.FIRST COME FIRST SERVE ALGORITHM:

AIM:

ALGORITHM:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int RQ[100],i,n,TotalHeadMoment=0,initial;
printf("Enter the number of Requests\n");
scanf("%d",&n);
printf("Enter the Requests sequence\n");
for(i=0;i<n;i++)
scanf("%d",&RQ[i]);
printf("Enter initial head position\n");
scanf("%d",&initial);
for(i=0;i<n;i++)
{
TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
}
printf("Total head moment is %d",TotalHeadMoment);
return 0;
}
```

OUTPUT:

Enter the number of Request
8
Enter the Requests Sequence
95 180 34 119 11 123 62 64
Enter initial head position
50
Total head movement is 644

Example: Given the following queue -- 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199.

RESULT:

EX NO: 15.B

DATE:

SHORTEST SEEK TIME FIRST (SSTF) ALGORITHM

AIM:

ALGORITHM:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;
printf("Enter the number of Requests\n");
scanf("%d",&n);
printf("Enter the Requests sequence\n");
for(i=0;i<n;i++)
scanf("%d",&RQ[i]);
printf("Enter initial head position\n");
scanf("%d",&initial);
while(count!=n)
{
int min=1000,d,index;
for(i=0;i<n;i++)
{
d=abs(RQ[i]-initial);
if(min>d)
{min=d;
index=i;
}
}
TotalHeadMoment=TotalHeadMoment+min;
initial=RQ[index];
// 1000 is for max
// you can use any number
RQ[index]=1000;
count++;
}
printf("Total head movement is %d",TotalHeadMoment);
return 0;
}
```

OUTPUT:

```
Enter the number of Request
8
Enter Request Sequence
95 180 34 119 11 123 62 64
Enter initial head Position
50
Total head movement is 236
```

RESULT:

EX NO: 15.C

DATE:

C-SCAN DISK SCHEDULING ALGORITH

AIM:

ALGORITHM:

EX NO:	
DATE:	

PROGRAM:

```
#include<stdio.h>
main()
{
int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;
//clrscr();
printf("enter the no of tracks to be traveresed"); scanf("%d'",&n);
printf("enter the position of head"); scanf("%d",&h);
t[0]=0;t[1]=h;
printf("enter  total  tracks"); scanf("%d",&tot);
t[2]=tot-1;
printf("enter  the  tracks"); for(i=3;i<=n+2;i++)
scanf("%d",&t[i]); for(i=0;i<=n+2;i++)
for(j=0;j<=(n+2)-i-1;j++)
if(t[j]>t[j+1])
{
for(i=0;i<=n+2;i++) if(t[i]==h);
j=i;
break;
temp=t[j]; t[j]=t[j+1]; t[j+1]=temp;
}
p=0;
while(t[j]!=tot-1)
{
atr[p]=t[j]; j++;
p++;
}
atr[p]=t[j]; p++;
i=0;
while(p!=(n+3) && t[i]!=t[h])
{
atr[p]=t[i]; i++; p++;
}

for(j=0;j<n+2;j++)
{
if(atr[j]>atr[j+1])
d[j]=atr[j]-atr[j+1];
else
d[j]=atr[j+1]-atr[j];
sum+=d[j];
}
printf("total header movements%d",sum); printf("avg is %f",(float)sum/n);
getch();
```

EX NO:	
DATE:	

}

OUTPUT:

RESULT

EX NO: 16

DATE:

INSTALL ANY GUEST OPERATING SYSTEM LIKE LINUX USING VMWARE

AIM:

Use a PC That Supports Virtualization

A Virtual machine is a software environment that replicates the conditions of a hardware environment : a personal computer . The environment is based on the hardware of physical PC and limited only by the components within.

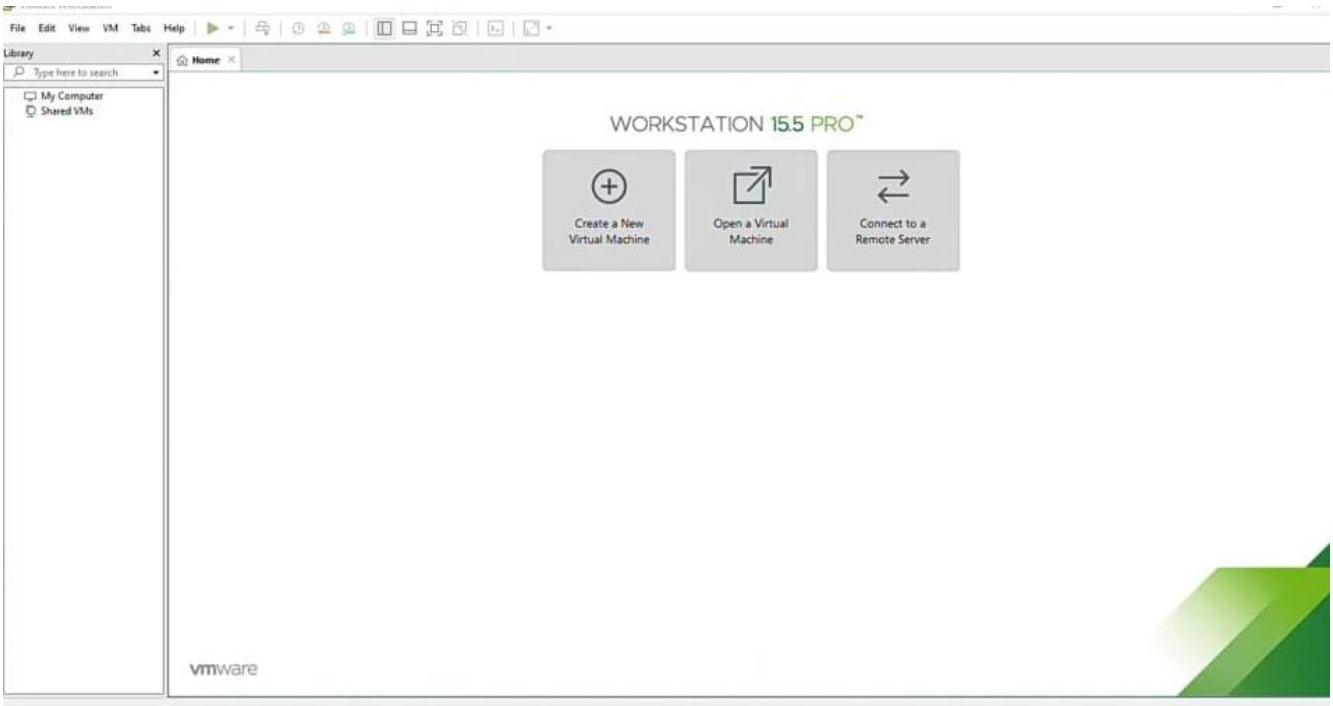
Several VM tools make it esay to install Linux operating systems (OS). **VMware** produces the most accomplised virtual machine applications.

Install Vmware Workstation Player

STEP 1 : To start , head to the **VMware website** and download the latest version of their Workstation Player tool . Here VMware Workstation 15 Players(150 MB) is downloaded . Vmware Workstation Player includes everything that is needed for standard virtual machine tasks.

STEP 2 : launch the installer and follow the installation wizard. Select the option to install an “Enhanced Keyborded Driver”.

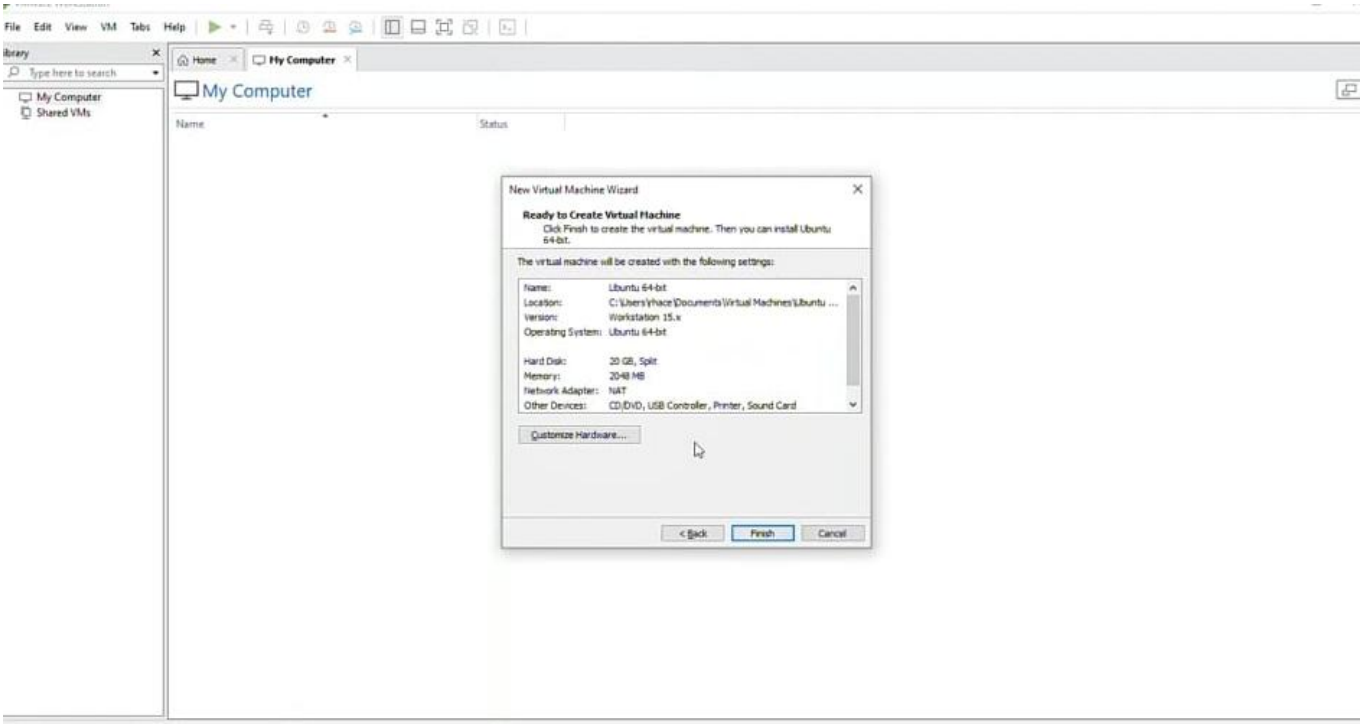
STEP 3: Proceed through the installation wizard , and restart windows when prompted.

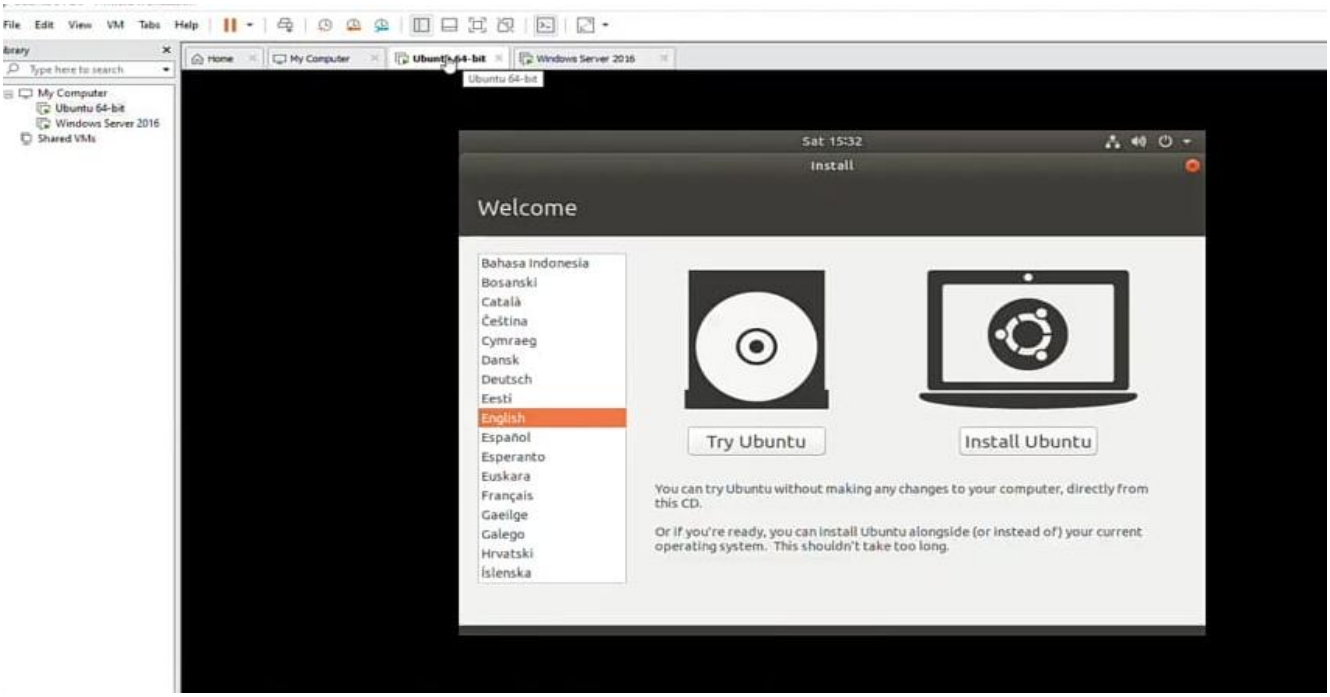


EX NO:	
DATE:	

Choose Your preferred Linux OS

linux OS like Linux distros is **suited to running in a VM**. All 32-bit and 64-bit distros work in a virtual machine .





EX NO:	
DATE:	

Creating Linux Virtual Machine

- Start by launching Vmware Workstation player . When you're ready to create a VM:

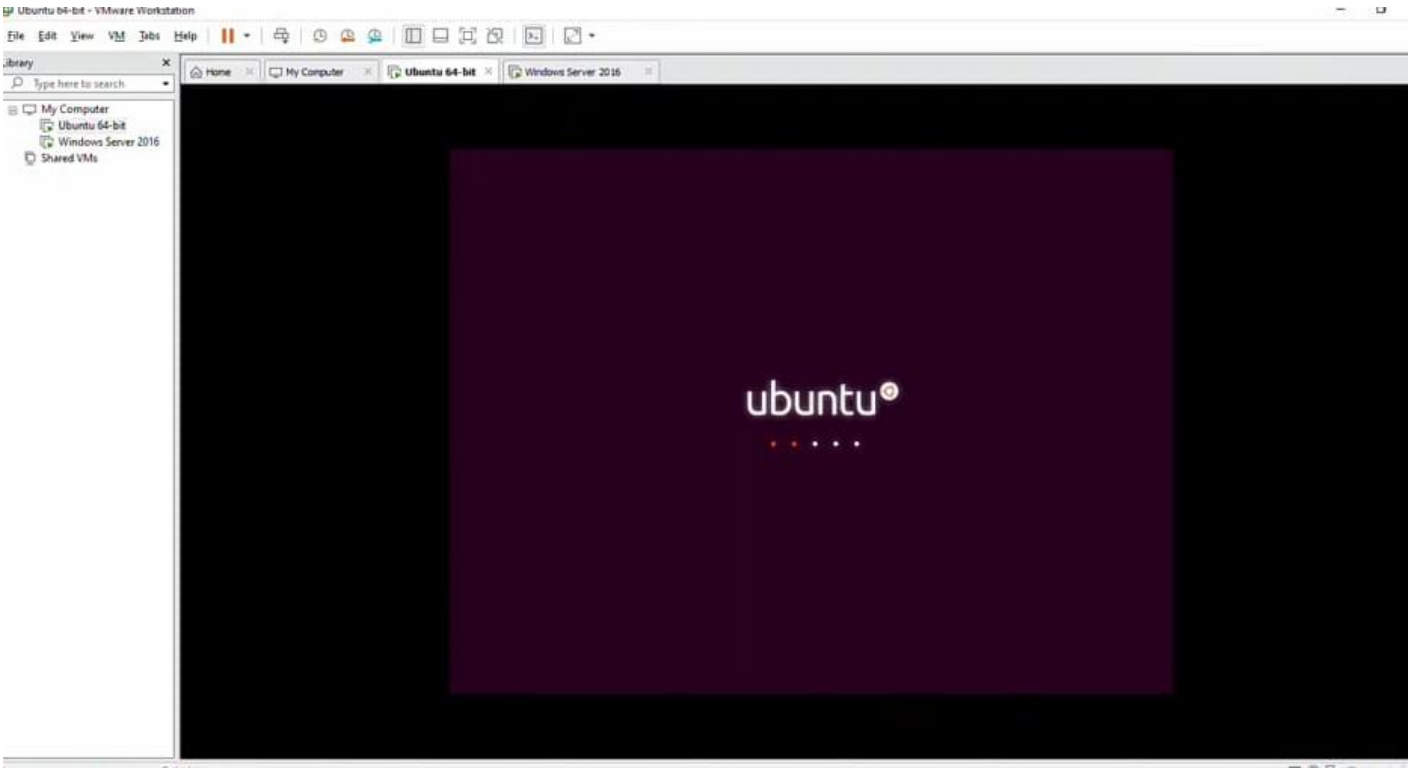
1. Click **Create a New Virtual Machine**
2. Select the default option , **Installer disc image file(iso)**
3. Click Browse to find the ISO file
4. With "guest" OS selected, Click Next
5. Select Linux as the Guest operating system type
6. Under Version , scroll through the list and select the OS
7. Click Next to proceed and if necessary, input a Virtual machine name
8. Confirm the storage location and change if needed

After selecting and configuring the operating system , build the virtual machine.

1. Under Specify disk Capacity adjust maximum disk size if required (default is enough)
2. Selection split Virtual disk into multiple files as this makes moving the VM to a new PC easy
3. Click Next then confirm the details on the next screen
4. If anything seems wrong click Back, otherwise click Finish.
    - Now the Linux virtual machine will be added to VMware Workstation player .
    - To installing Linux in Vmware is follow the steps given below :

1. Download the free Vmware Workstation player .
2. Install , and reset windows
3. Create and configure your virtual machine
4. install Linux in the virtual machine
5. Restart the virtual machine and use Linux

EX NO:	
DATE:	



RESULT: