Practical No. 1

Aim :- Execute the Disk Operating System (DOS) commands.

Theory :- DOS Commands are the commands available in MS-DOS that are used to interact with the operating system and other command-based software. Unlike in windows, DOS commands are the primary way in which you use the OS. Windows and other modern OSs use a graphics-based system designed for touch or a mouse.

MS-DOS Commands :-

i) Date - The date command is used to change or show the current date.

ii) Time - To find the current time, MS-DOS provide the TIME command.

iii) Cd - CD (change directory) is a command used to switch directories in MS-DOS

iv) Md - MD (make directory) is used to create your own directory in the specified drive.

v) Start - To open your folder using this command.

Start directory_name.

vi) Rd - Rd (Remove directory) is used to remove any directory from any drive.

rd directory name.

vii) Path - Path is used to specify the location where MS-DOS looks.

viii) Chkdsk - It is a utility which checks the computer's hard drives status.

ix) Copy - It allows you to copy one or more files to an alternative location.

x) Xcopy - xcopy is a powerful version of the copy command with some additional features.

xi) Format - Format is used to erase information of a computer diskette.

xii) Diskpart - Diskpart is used to delete and create partitions on the hard drive.

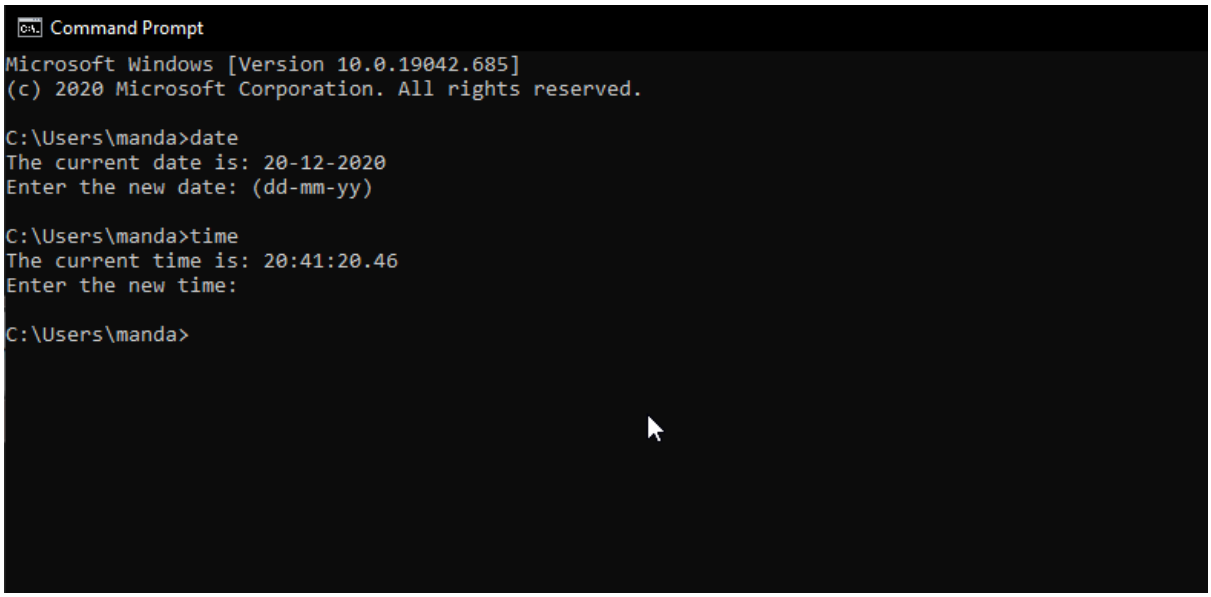
xiii) Del - del is used to delete files from the computer.

xlv) move - move allows you to move files or directories from one folder to another or from one drive to another.

xv) Exit - exit command is used to end the command session that you're working on.

xvi) cls - cls is used to clear screen in command prompt.

Conclusion :- We have successfully executed the Disk Operating System commands.

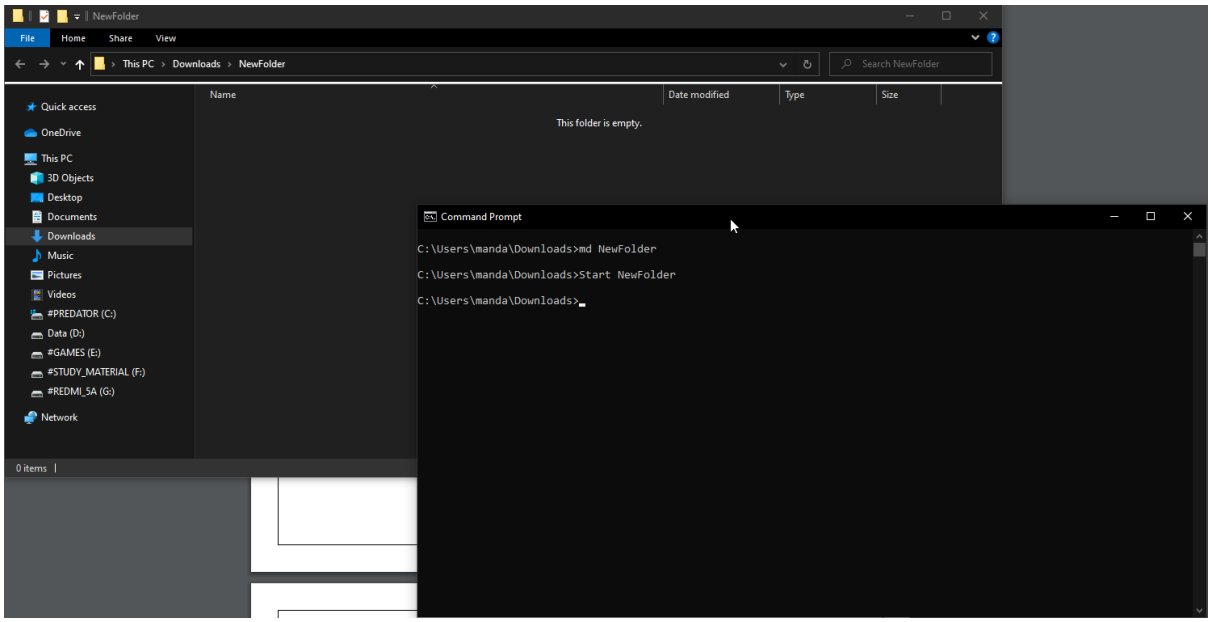


```
Command Prompt
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\manda>cd Downloads

C:\Users\manda\Downloads>cd..

C:\Users\manda>
```




```
Command Prompt

C:\Users\manda>path
PATH=d:/ansel

C:\Users\manda>path c:/Java

C:\Users\manda>path
PATH=c:/Java

C:\Users\manda>
```

Practical No. 2

Aim :- Install and configure windows 9x, windows NT, windows 2000 & windows XP operating systems.

Theory :-

- Installation :-

i> Windows 9x -

1) Insert the windows 9x startup disk in the floppy disk drive and then restart your PC.

2) When the windows startup menu is displayed, choose the start computer with CD-ROM support option and press ENTER.

3) If CD-ROM support is provided by the generic drivers on the startup disk. You will receive one of the following message, where x is the drive letter that is assigned to your CD-ROM drive : Drive x:= Driver MSCD001
Drive x:= Driver DEMCD001

4) Insert the windows 9x CD-ROM in the drive type the following command at a command prompt and then press ENTER X:/ setup

5) When you receive the message, press ENTER and follow instruction to complete setup procedure, It will start routine check-up on your system, to continue press ENTER.

ii) Windows NT

1) Insert your windows NT 4.0 CD and disk 1 of your 3NT workstations setup disk.

2) After this boot up system then windows NT will detect your hardware configuration.

3) Soon you will be promted to insert NT setup disk 2 & after that inserting hit ENTER.

4) After this, search for storage device and insert disk 3 of your windows NT workstat -ion setup.

5) Windows will detect a CD-ROM, then click on continue components >> Standard settings >> partitions.

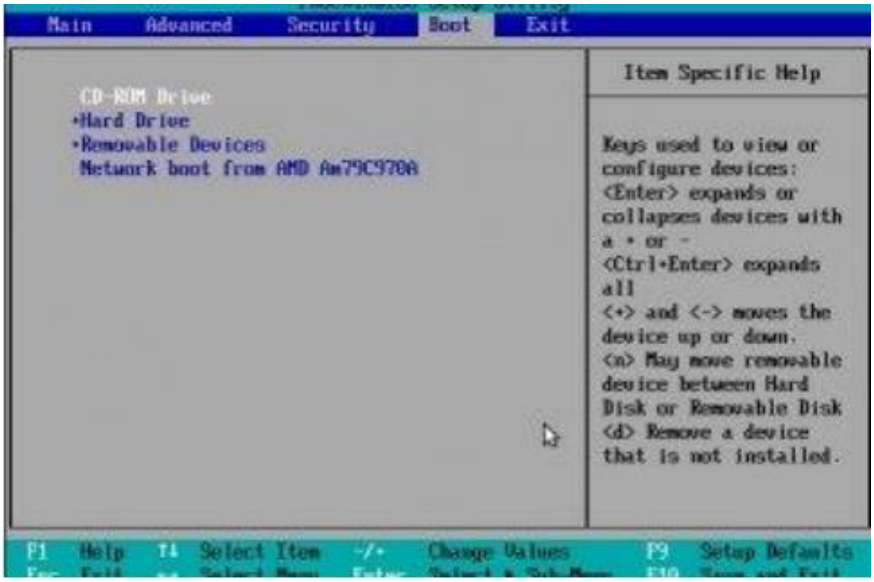
6) After partitions, you will be asked where you want to install your Operating System.

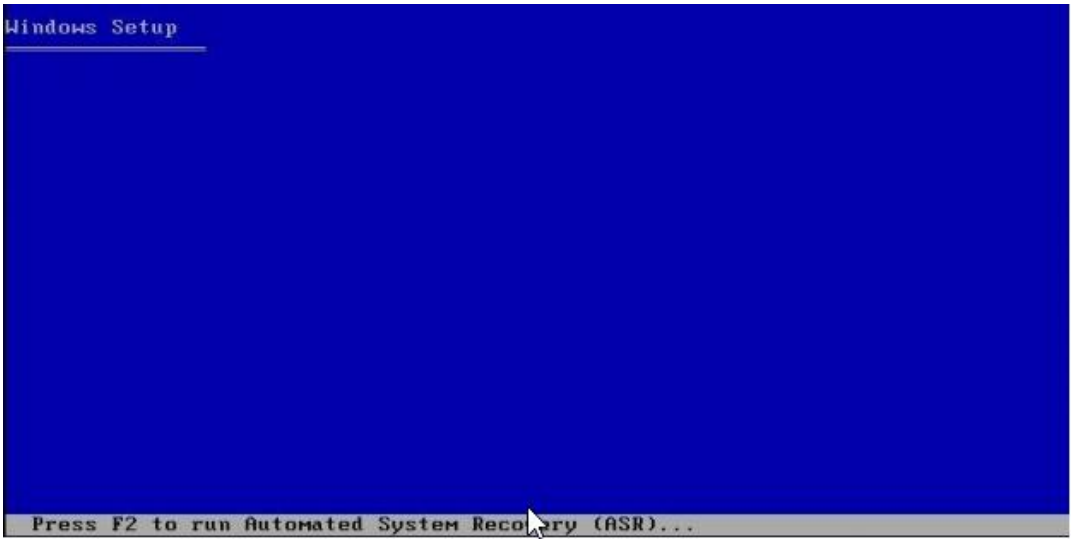
7) After selecting location, hit ENTER and reboot your device after installation.

•Configuration :-

i) Remove floopy disk or CDs from drive, Click on setup.
ii) Select User as per your choice, then fill your details and press ENTER.
iii) Enter cirtificate of Authentication & OEM no.
iv) Now enter name for your computer.
v) Set password for admin.
vi) select defaut components and press Enter key.
vii) For Internet Connection, NT networks prompt will appear, leave it for now and press NEXT.
viii) Select your timezone and Language, then press FINISH option and restart your computer.
ix) After rebooting, a login page will appear, click on it and enter password which have you set earlier and your windows is now successfully installed.

Conclusion :- We have successfully Installed and configured windows 9x, windowsNT, windows 2000 and windows XP operating system.

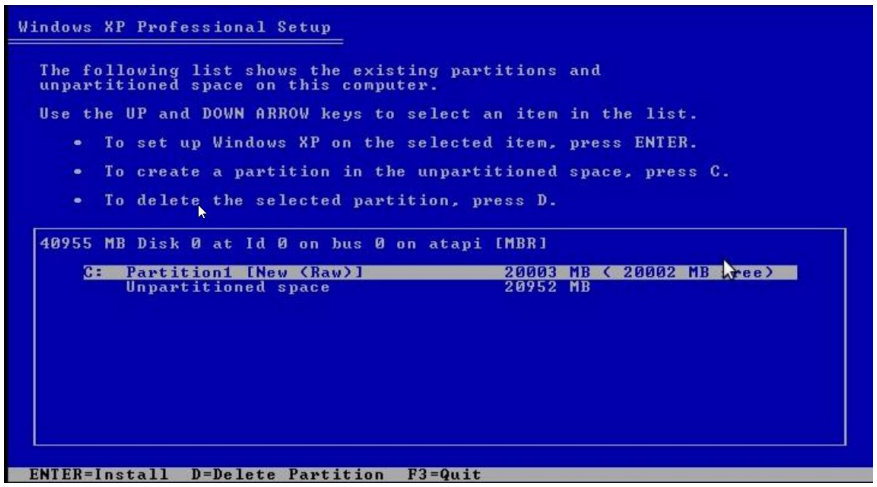

Main
Advanced
Security
Boot
Exit
CD-ROM Drive
•Hard Drive
•Removable Devices
Network boot from AMD Am79C970A
Item Specific Help
Keys used to view or configure devices:
<Enter> expands or collapses devices with a + or -
<Ctrl+Enter> expands all
<+> and <-> moves the device up or down.
<n> May move removable device between Hard Disk or Removable Disk
<d> Remove a device that is not installed.
F1 Help
↑↓ Select Item
-/+ Change Values
F9 Setup Defaults

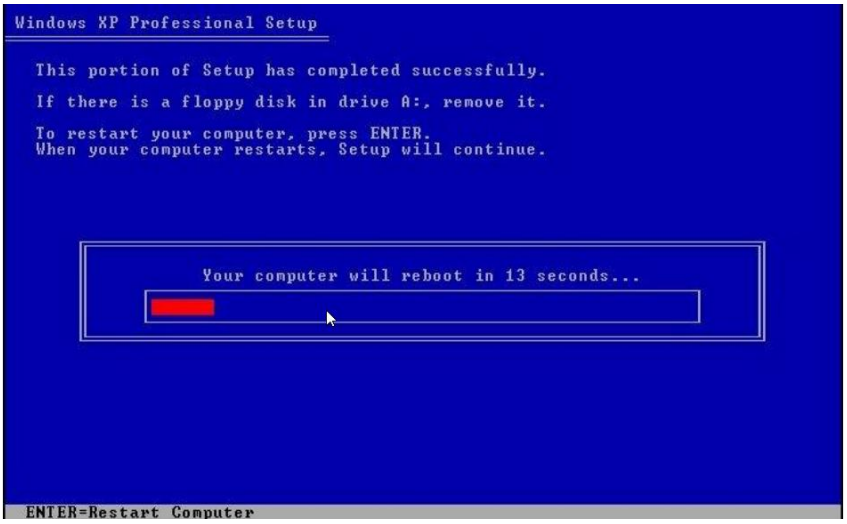

Windows Setup
Press F2 to run Automated System Recovery (ASR)...


Windows XP Professional Setup
Welcome to Setup.
This portion of the Setup program prepares Microsoft(R) Windows(R) XP to run on your computer.
• To set up Windows XP now, press ENTER.
• To repair a Windows XP installation using Recovery Console, press R.
• To quit Setup without installing Windows XP, press F3.
ENTER=Continue R=Repair F3=Quit


Windows XP Professional Setup
The following list shows the existing partitions and unpartitioned space on this computer.
Use the UP and DOWN ARROW keys to select an item in the list.
• To set up Windows XP on the selected item, press ENTER.
• To create a partition in the unpartitioned space, press C.
• To delete the selected partition, press D.
40955 MB Disk 0 at Id 0 on bus 0 on atapi [MBR]
Unpartitioned space 40955 MB
ENTER=Install C=Create Partition F3=Quit

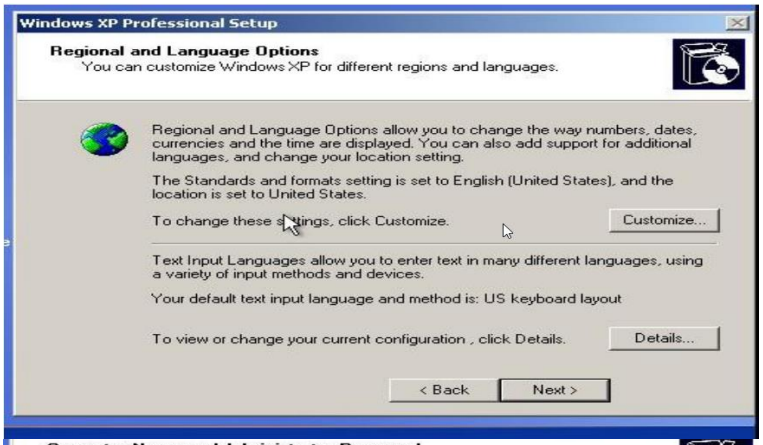

Windows XP Professional Setup
You asked Setup to create a new partition on 40955 MB Disk 0 at Id 0 on bus 0 on atapi [MBR].
• To create the new partition, enter a size below and press ENTER.
• To go back to the previous screen without creating the partition, press ESC.
The minimum size for the new partition is 8 megabytes (MB).
The maximum size for the new partition is 40947 megabytes (MB).
Create partition of size (in MB): 20000
ENTER=Create ESC=Cancel


Windows XP Professional Setup
The following list shows the existing partitions and unpartitioned space on this computer.
Use the UP and DOWN ARROW keys to select an item in the list.
• To set up Windows XP on the selected item, press ENTER.
• To create a partition in the unpartitioned space, press C.
• To delete the selected partition, press D.
40955 MB Disk 0 at Id 0 on bus 0 on atapi [MBR]
C: Partition1 [New (Raw)] 20003 MB (20002 MB free)
Unpartitioned space 20952 MB
ENTER=Install D=Delete Partition F3=Quit

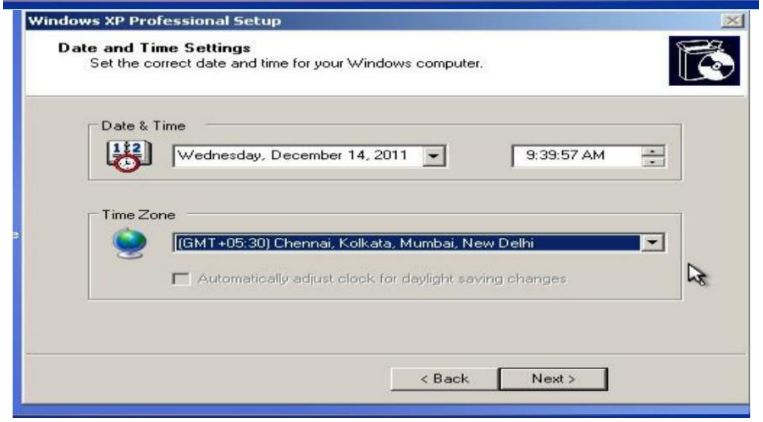

Windows XP Professional Setup
This portion of Setup has completed successfully.
If there is a floppy disk in drive A:, remove it.
To restart your computer, press ENTER.
When your computer restarts, Setup will continue.
Your computer will reboot in 13 seconds...
ENTER=Restart Computer

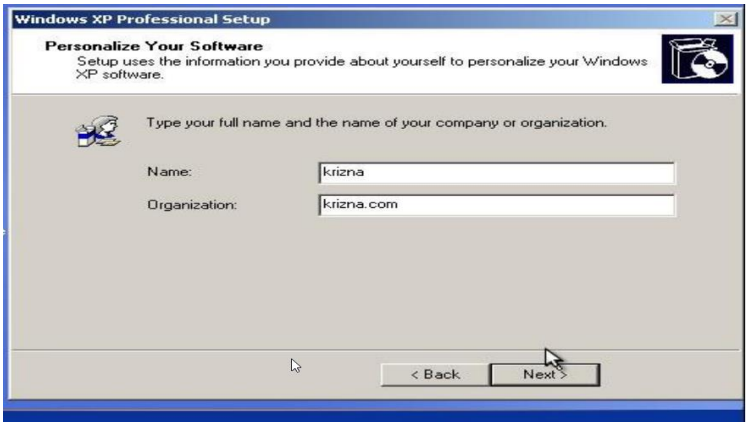

Windows XP Professional Setup
The partition you selected is not formatted. Setup will now format the partition.
Use the UP and DOWN ARROW keys to select the file system you want, and then press ENTER.
If you want to select a different partition for Windows XP, press ESC.
Format the partition using the NTFS file system (Quick)
Format the partition using the FAT file system (Quick)
Format the partition using the NTFS file system
Format the partition using the FAT file system
ENTER=Continue ESC=Cancel

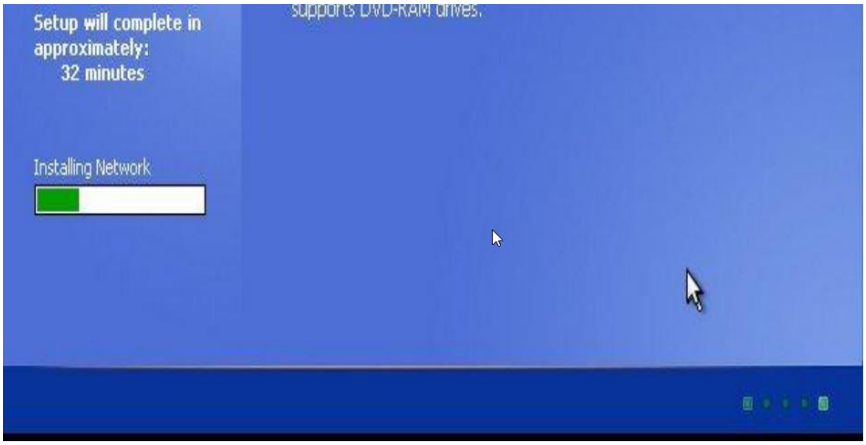

Windows xp
Collecting information
Dynamic Update
Preparing installation
Installing Windows
Finalizing installation
Setup will complete in approximately:
37 minutes
Installing Devices
Try the easiest Windows® yet
Windows XP makes it easy to manage all of your information. We've enhanced the My Documents, My Pictures, and My Music folders to make them more useful. Now whenever you open these folders, you'll also see handy shortcuts to the most common tasks for documents, pictures, and music.
Integrated support for Web publishing means that you can put your documents on the Internet and get to them easily from any location.
Windows XP Professional also supports integrated CD recording, so now you can easily save files to a CD-R or CD-RW drive.


Windows XP Professional Setup
Regional and Language Options
You can customize Windows XP for different regions and languages.
Regional and Language Options allow you to change the way numbers, dates, currencies and the time are displayed. You can also add support for additional languages, and change your location setting.
The Standards and formats setting is set to English (United States), and the location is set to United States.
To change these settings, click Customize.
Customize...
Text Input Languages allow you to enter text in many different languages, using a variety of input methods and devices.
Your default text input language and method is: US keyboard layout
To view or change your current configuration , click Details.
Details...
< Back
Next >


Computer Name and Administrator Password
You must provide a name and an Administrator password for your computer.
Setup has suggested a name for your computer. If your computer is on a network, your network administrator can tell you what name to use.
Computer name:
KRIZNA-66E42DB2
Setup creates a user account called Administrator. You use this account when you need full access to your computer.
Type an Administrator password.
Administrator password:
Confirm password:
< Back
Next >


Windows XP Professional Setup
Date and Time Settings
Set the correct date and time for your Windows computer.
Date & Time
Wednesday, December 14, 2011
9:39:57 AM
Time Zone
(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi
Automatically adjust clock for daylight saving changes
< Back
Next >


Windows XP Professional Setup
Personalize Your Software
Setup uses the information you provide about yourself to personalize your Windows XP software.
Type your full name and the name of your company or organization.
Name:
krizna
Organization:
krizna.com
< Back
Next >


Setup will complete in approximately:
32 minutes
Installing Network


Windows xp
Welcome to Microsoft Windows
Thank you for purchasing Microsoft Windows XP.
Let's spend a few minutes setting up your computer.
Welcome to Windows XP!
To continue, click Next.
Next


Windows xp
Who will use this computer?
Type the name of each person who will use this computer. Windows will create a separate user account for each person so you can personalize the way you want Windows to organize and display information, protect your files and computer settings, and customize the desktop.
Your name:
admin
2nd User:
3rd User:
4th User:
5th User:
These names will appear on the Welcome screen in alphabetical order. When you start Windows, simply click your name on the Welcome screen to begin. If you want to set passwords and limit permissions for each user, or add more user accounts after you finish setting up Windows, just click Control Panel on the Start menu, and then click User Accounts.
For help, click here or press F1.
Back
Next

Practical No. 3.

Aim :- Execute the LINUX command: man, cyppropos, clear, ls, mkdir, cd, rmdir, pwd, rm, touch, mv, tr, wc, sort, grep, wall, write, who, chmod, useradd, usermod, kill, ssh, ftp, telnet.

Theory :- The function and syntax of each command mentioned as given below :

- man - to display User manual of any command just type man and next to it the command which you want to see

- apropos - it finds exact command from the keywords
  syntax :- $ cyppropos [keyword.]

- clear - This command is used to clear terminal screen
  syntax :- $ clear

- ls - ls is a command-line utility for listing the contents of a directory or directories given to it via standard input.

- mkdir - It used to create new directories
  syntax :- $ mkdir [directory_name.]

- wc - wc is used to count number of lines of given files,
  syntax $ touch [Filename] | wc 110811
  $ wc [File_name]
- sort - sort is used to arrange records in particular order
  syntax :- $ sort [File_name]
- grep - grep is used to search specific string of character in specific file or directory
  syntax :- $ grep string_name File_name
- wall - it is used to write message to all other users.
- write - write is used to send message to other user.
- who - who is used to find time of last boot, logged in users, remote host name of user, run level of system and other information
- chmod - chmod is used to change the properties or access of the file.
  syntax :- $ chmod [options] [File_name]
- useradd - useraddd allows you to add new users to system
  syntax :- $ useradd [option] [name of user]

- cd - It is used to change the current working directory in linux.
  syntax :- $ cd [directory_name]
  "or"
  $ cd..

- rmdir - rmdir is used to remove empty directories from the filesystem.
  syntax :- $rmdir directory_name.

- pwd - pwd stands for print working directory and prints the path of the working directory.

- rm - this command is used to remove objects such as : files, symbolic links from the file system
  Syntax :- $rm [File_name]

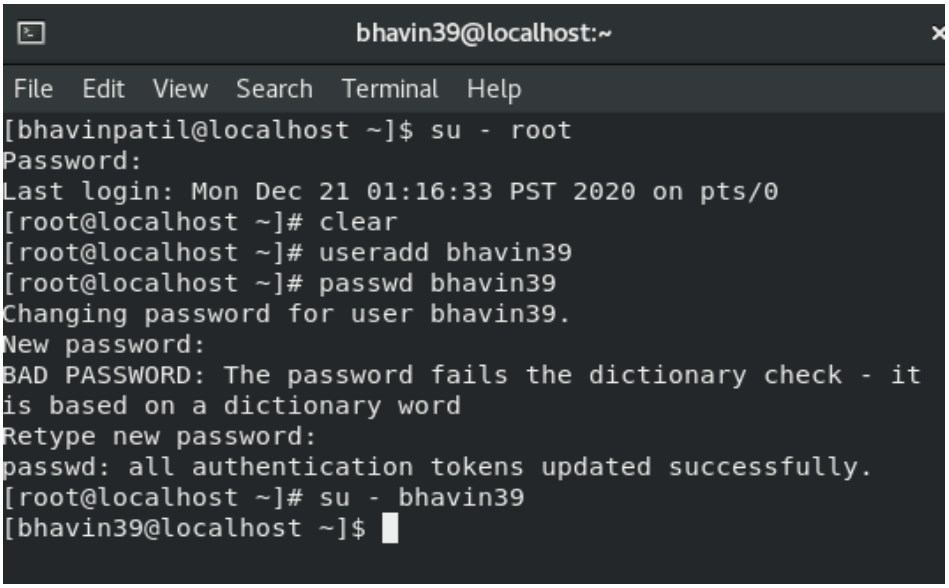
- touch - touch is used to create, change or modify timepan of a file
  syntax :- $ touch [file_name]

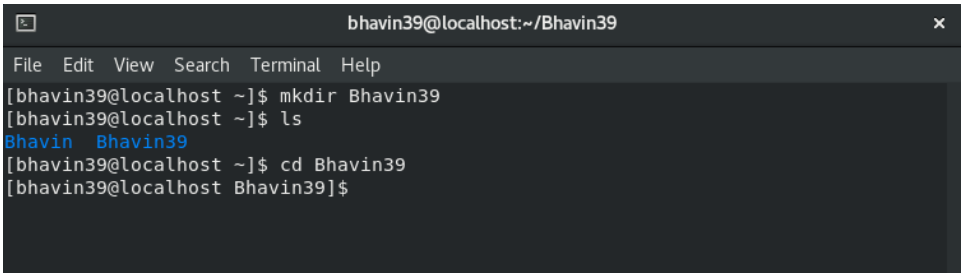
- mv - mv is used to move one or more files from one location to others.
  syntax :- $ mv [filename] [location]

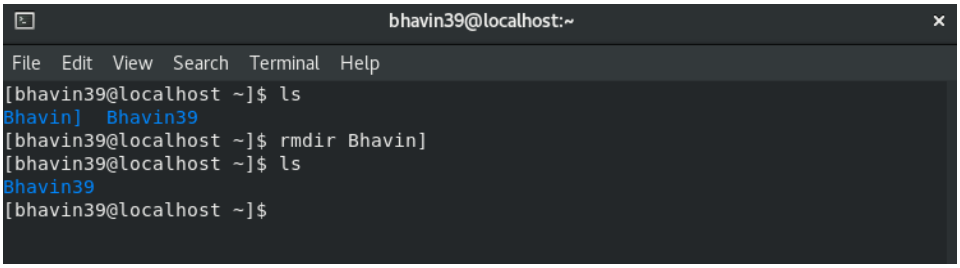
- tr - tr is command-line utility to translate or delete character
  syntax :- $ tr [set 1] [set 2]

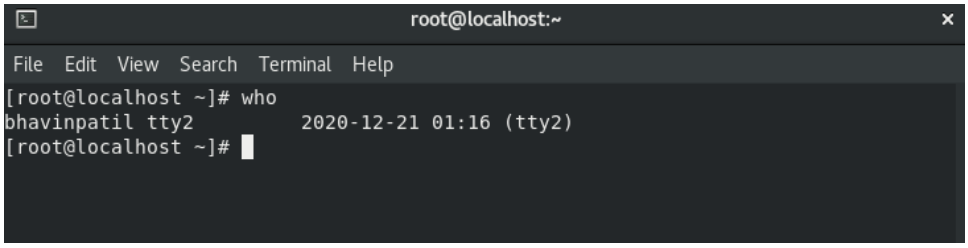
- usermod - usermod modifys user
  syntax :- $ usermod [option] [user]
- kill - It is used to sends signals to processor which terminates the process
  syntax :- $ kill
- ssh - ssh stands for secure shell, it is used to securely connect user to remote server / system
  syntax :- $ ssh [IP address]
- ftp - ftp is used to transfer files to form a remote network
  syntax : $ ftp [server_name]
- telnet - It is an old protocal used to connect remote system even TCP/IP network
  syntax :- $ telnet [server_IP_addr]

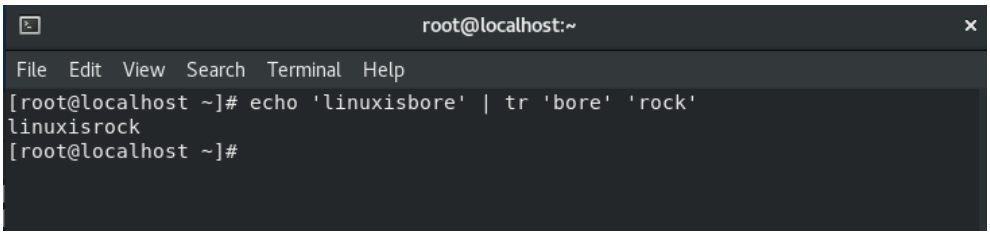
Conclusion :- Hence we have executed all the mentioned linux commands successfully.

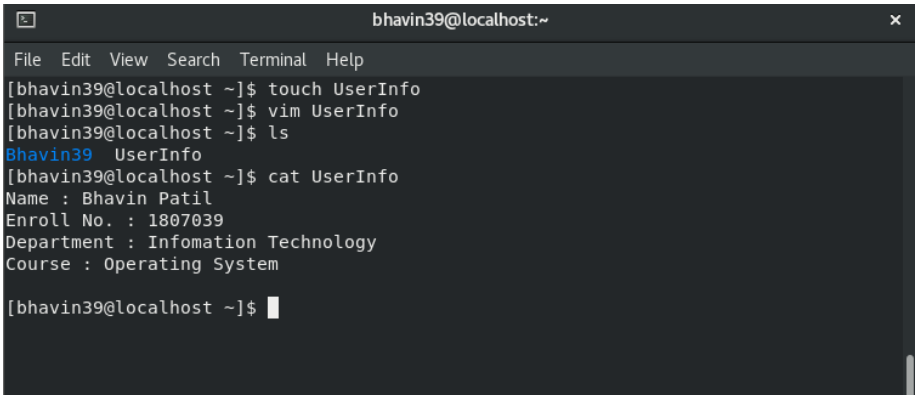

bhavin39@localhost:~
File Edit View Search Terminal Help
[bhavinpatil@localhost ~]$ su - root
Password:
Last login: Mon Dec 21 01:16:33 PST 2020 on pts/0
[root@localhost ~]# clear
[root@localhost ~]# useradd bhavin39
[root@localhost ~]# passwd bhavin39
Changing password for user bhavin39.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]# su - bhavin39
[bhavin39@localhost ~]$

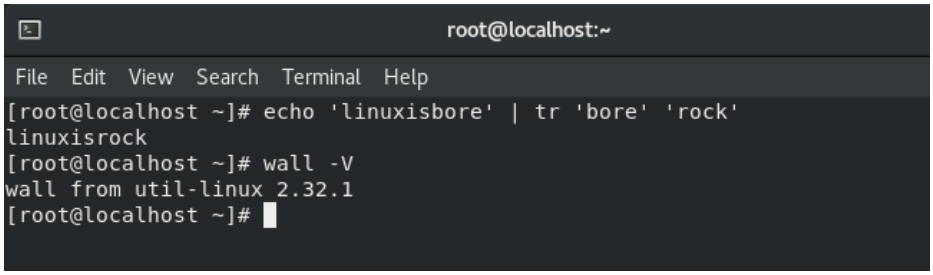

bhavin39@localhost:~/Bhavin39
File Edit View Search Terminal Help
[bhavin39@localhost ~]$ mkdir Bhavin39
[bhavin39@localhost ~]$ ls
Bhavin Bhavin39
[bhavin39@localhost ~]$ cd Bhavin39
[bhavin39@localhost Bhavin39]$

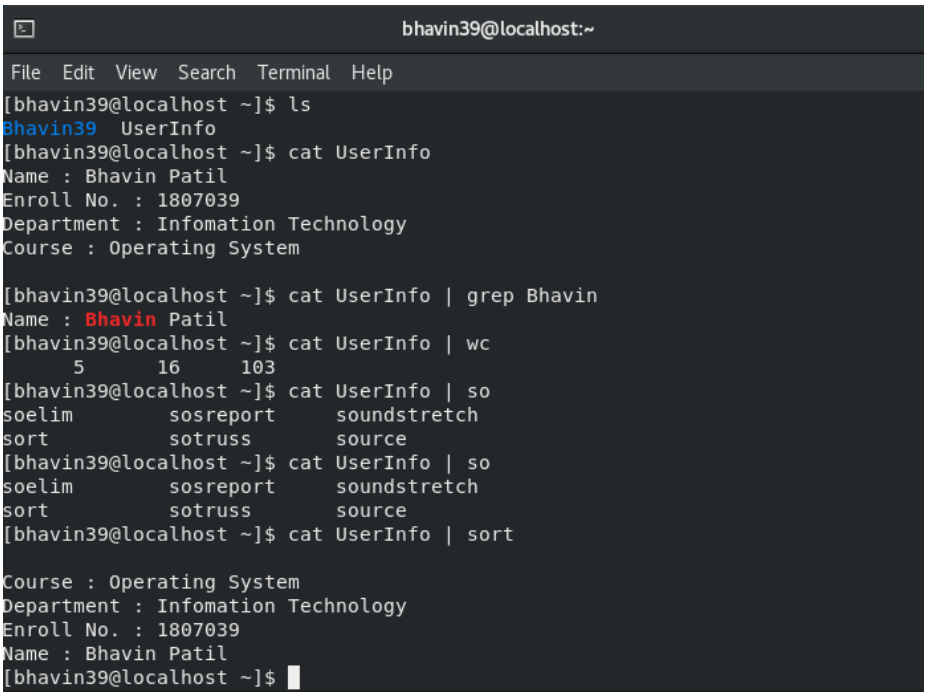

bhavin39@localhost:~
File Edit View Search Terminal Help
[bhavin39@localhost ~]$ ls
Bhavin] Bhavin39
[bhavin39@localhost ~]$ rmdir Bhavin]
[bhavin39@localhost ~]$ ls
Bhavin39
[bhavin39@localhost ~]$

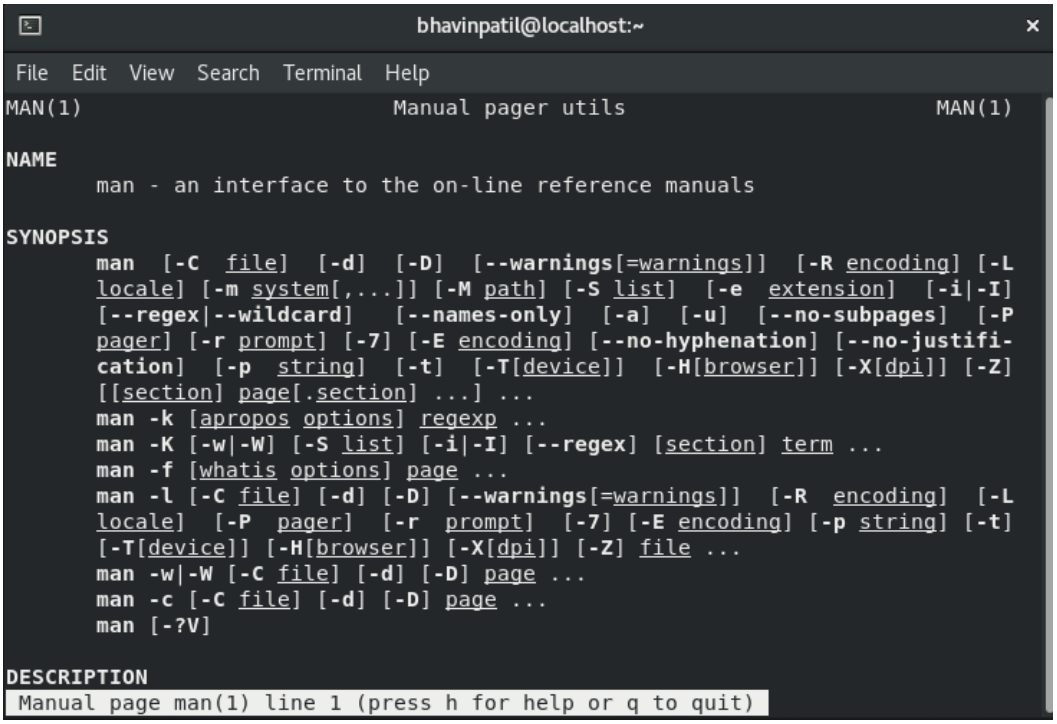

root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# who
bhavinpatil tty2 2020-12-21 01:16 (tty2)
[root@localhost ~]#

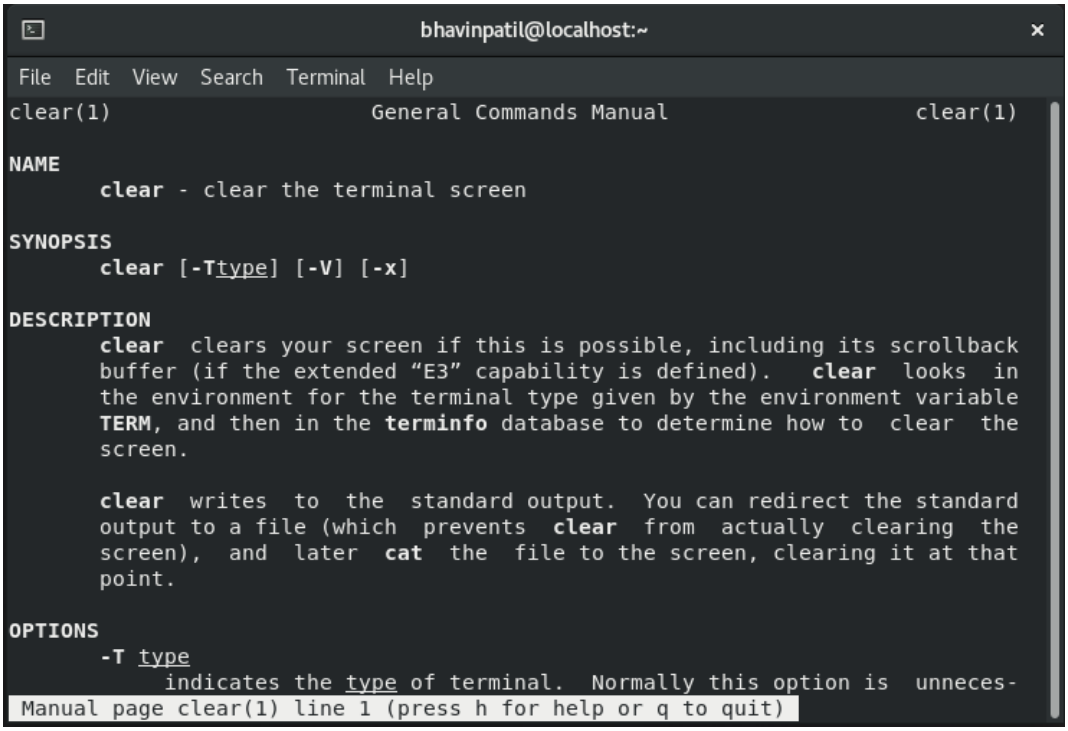

root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# echo 'linuxisbore' | tr 'bore' 'rock'
linuxisrock
[root@localhost ~]#


bhavin39@localhost:~
File Edit View Search Terminal Help
[bhavin39@localhost ~]$ touch UserInfo
[bhavin39@localhost ~]$ vim UserInfo
[bhavin39@localhost ~]$ ls
Bhavin39 UserInfo
[bhavin39@localhost ~]$ cat UserInfo
Name : Bhavin Patil
Enroll No. : 1807039
Department : Infomation Technology
Course : Operating System
[bhavin39@localhost ~]$


root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# echo 'linuxisbore' | tr 'bore' 'rock'
linuxisrock
[root@localhost ~]# wall -V
wall from util-linux 2.32.1
[root@localhost ~]#


bhavin39@localhost:~
File Edit View Search Terminal Help
[bhavin39@localhost ~]$ ls
Bhavin39 UserInfo
[bhavin39@localhost ~]$ cat UserInfo
Name : Bhavin Patil
Enroll No. : 1807039
Department : Infomation Technology
Course : Operating System
[bhavin39@localhost ~]$ cat UserInfo | grep Bhavin
Name : Bhavin Patil
[bhavin39@localhost ~]$ cat UserInfo | wc
5 16 103
[bhavin39@localhost ~]$ cat UserInfo | so
soelim sosreport soundstretch
sort sotruss source
[bhavin39@localhost ~]$ cat UserInfo | so
soelim sosreport soundstretch
sort sotruss source
[bhavin39@localhost ~]$ cat UserInfo | sort
Course : Operating System
Department : Infomation Technology
Enroll No. : 1807039
Name : Bhavin Patil
[bhavin39@localhost ~]$


bhavinpatil@localhost:~
File Edit View Search Terminal Help
MAN(1) Manual pager utils MAN(1)
NAME
man - an interface to the on-line reference manuals
SYNOPSIS
man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
man -f [whatis options] page ...
man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
man -w|-W [-C file] [-d] [-D] page ...
man -c [-C file] [-d] [-D] page ...
man [-?V]
DESCRIPTION
Manual page man(1) line 1 (press h for help or q to quit)


bhavinpatil@localhost:~
File Edit View Search Terminal Help
clear(1) General Commands Manual clear(1)
NAME
clear - clear the terminal screen
SYNOPSIS
clear [-Ttype] [-V] [-x]
DESCRIPTION
clear clears your screen if this is possible, including its scrollback buffer (if the extended "E3" capability is defined). clear looks in the environment for the terminal type given by the environment variable TERM, and then in the terminfo database to determine how to clear the screen.
clear writes to the standard output. You can redirect the standard output to a file (which prevents clear from actually clearing the screen), and later cat the file to the screen, clearing it at that point.
OPTIONS
-T type
indicates the type of terminal. Normally this option is unneces-
Manual page clear(1) line 1 (press h for help or q to quit)

Practical No. 4

Aim :- Develop, debug and execute a C program to simulate the FCFS CPU scheduling algorithm to find turnaround time and waiting time.

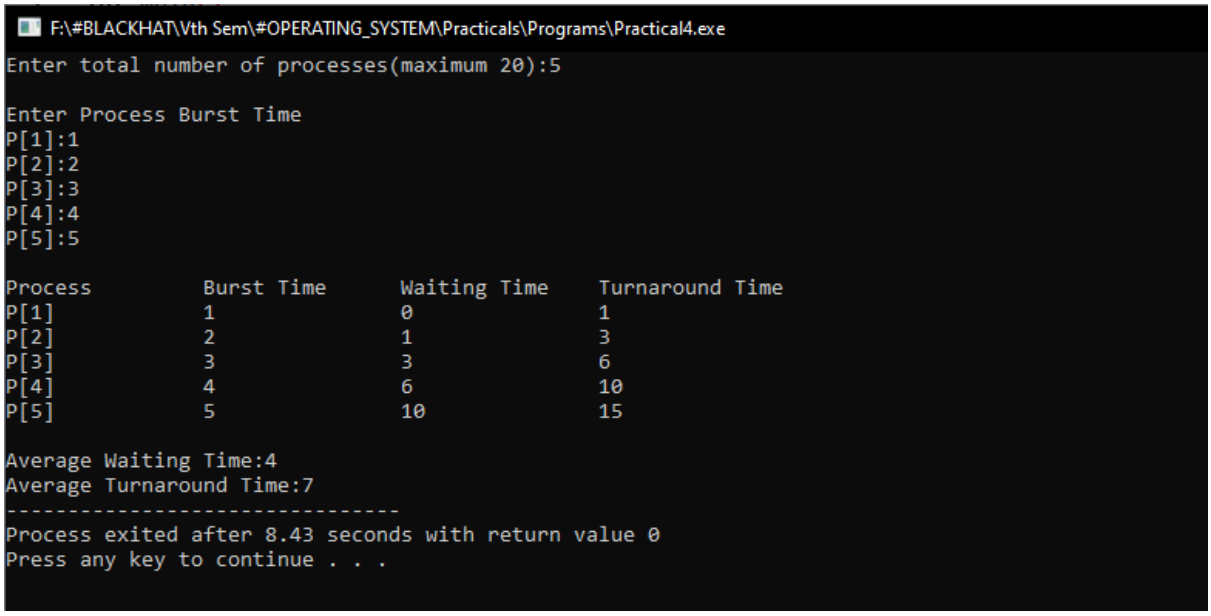
Theory :- In the First come First serve scheduling algorithm, as the name suggests, the process which arrives First, gets executed First, This is used in Batch systems. It's easy to understand and implement programmatically, using a queue data structure, where a new process enters through the tail and schedule selects process from the head of the queue. A perfect real like example of FCFS scheduling is buying tickets at ticket counter.

Program :-

```
#include <stdio.h>
int main() {
    int n, BurstTime[20], bt[20], wt[20], tat[20], avwt[
    i, j, avwt = 0, avtat = 0;
    printf("Enter total number of processes");
    scanf("%d", &n);
    printf("\n Enter Process Burst Time : ");
        for (i = 0; i < n; i++) {
            printf("P[%d]", i+1);
            scanf("%d", &bt[i]);
        }
        wt[0] = 0;
        for (i = 1; i < n; i++) {
```

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

Conclusion :- Hence, we have developed, debugged and executed a C program to schedule the FCFS CPU scheduling algorithm to find turnaround time and waiting time.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical4.exe
Enter total number of processes(maximum 20):5
Enter Process Burst Time
P[1]:1
P[2]:2
P[3]:3
P[4]:4
P[5]:5
Process Burst Time Waiting Time Turnaround Time
P[1] 1 0 1
P[2] 2 1 3
P[3] 3 3 6
P[4] 4 6 10
P[5] 5 10 15
Average Waiting Time:4
Average Turnaround Time:7
Process exited after 8.43 seconds with return value 0
Press any key to continue . . .

Practical No :- 5

Aim :- Develop, debug and execute a C-program to simulate the SJF CPU scheduling algorithm to find turnaround and waiting time.

Theory :- Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be premitive or non-premitive. It significantly reduces the average waiting time for other process awaiting execution.

There are two types of SJF : i) Premitive SJF
ii) Non-premitive SJF.

SJF algorithm can be premitive as well as non-premitive. Premitive scheduling is also known as short-remaining-time-first scheduling. In premetive approach, the new process arises when there is already executing process. If the burst of newly arriving process is lesser than the burst time of executing process then scheduller will prompt the execution of the process with lesser burst time.

Algorithm :- i. Declare pid, bt, art;

ii. function findaroundtime (process proc[], int n,
int wt[], int tat[])

for loop i=0; i<n; i++
set tat[i] = proc[i].bt + wt[i]

iii. function findwaitingtime (Process proc[], int n, int wt[])

declare rt[n]

for loop i=0 ; i<n ; i++

set rt[i] = proc[i].bt

set complete = 0 ; t = 0 ; mim = INT-MAX

set shortest = 0 ; finish-time

set boolean check = false

while loop (complete != n)

for loop ; j=0 ; j<n ; j++

IF (proc[j].art <= t) && (rt[j] < mim)

&& rt[i] > 0)

set mim = rt[i]

set shortest = j

set check = true

IF check == false

t++;

continue

rt[sortest]--;

set mim = rt[shortest]

IF mim == 0

set mim = INT_MAX

IF rt[shortest] == 0

complete ++;

set check = false

set finish-time = t+1

set wt[shortest] = finish-time - proc

[shortest].bt

- proc.[shortest].bt

IF wt[shortest] < 0

set wt[shortest] = 0

t ++;

iv Function findavgtime (process proc [], int n)

declare wt[n], tat[n], total_wt = 0,
total_tat = 0;

call findwaitingtime (proc, n, wt)
Find turnaroundtime (proc, n, wt, tat)

for loop i=0; i<n; i++

set total_wt = total_wt + wt[i]

set total_tat = total_tat + tat[i]

print proc[i].pid, proc[i].bt, wt[i], tat[i]

print averagewaitingtime i.e. total_wt

print averaverage turnaround time i.e. total_tat

v Funtion int main ()

declare process proc[] = {{1, 5, 1}, {2, 3, 1}, {3, 6, 2}, {4, 5, 3}}

set n = sizeof (proc) ;

call findavgtime (proc, n)

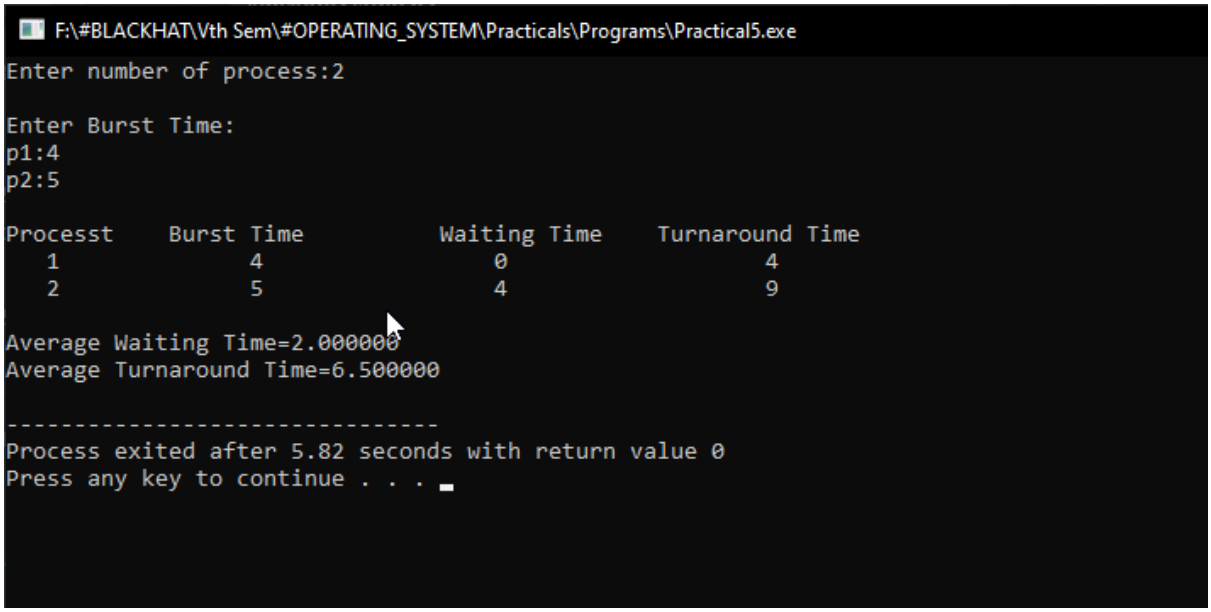
vi stop.

Program :-

```
#include <stdio.h>
#include <conio.h>
int main (){
    int   count, i, n, time, remain, flag=0,
          time_quantum;
    int   wait_time =0,
          turnaround_time = 0;
          at [10],
          bt [10],
          rt [10];
    printf("Enter total process :");
    scanf("%d", &n);
    remain = n;
    For( count=0; count<n; count++) {
        printf('Enter Arrival time and Burst
                time for process No. %d", count+1)
        scanF("%d", &at[count]);
        scanF("%d", &bt[count]);
          rt[count] = bt[count];
        }
    printf("Enter time Quantum : ");
    scanf("%d", &time_quantum);
    printf(" Process || Turnaround || Waiting Time")
    for( time=0, count=0; remain!=0) {
        if( rt[count] <= time_quantum &&
            rt[count > 0) {
            time += rt[count];
            rt[count] = 0;
            flag = 1;
        }
```

```
else if (rt[count] > 0) {
    rt[count] -= time_quantum;
    time += time_quantum;
}
if (rt[count] == 0 && flag == 1) {
    remain --;
    printf("P[%d] \t %d \t %d \n", count+1,
        time_at[count], time_at[count - bt
        [count]]);
    wait_time += time - at[count] - bt[count];
    termaround_time += time - at[count];
    flag = 0;
}
if (count == n-1)
    count = 0;
else if (at[count+1] <= time)
    count ++;
else
    count = 0;
}
printf("Average Waiting Time = %f", wait_time +
    1.0/n);
printf("Avg. Turnaround Time = %f", termaround_time *
    1.0/n);
return 0;
```

Conclusion :- Here, we have successfully developed and executed Shorted Job first CPU shedulsing program using C programing language.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical5.exe
Enter number of process:2
Enter Burst Time:
p1:4
p2:5
Processt Burst Time Waiting Time Turnaround Time
1 4 0 4
2 5 4 9
Average Waiting Time=2.000000
Average Turnaround Time=6.500000
Process exited after 5.82 seconds with return value 0
Press any key to continue . . .

Practical No. 6

Aim:- Develop and execute a C program to simulate the Round Robin CPU schedulling algorithm to find turnaround time and waiting time.

Theory :- A Round Robin is a CPU schedulling algorithm that shares equal portions of resource in a circular order to each process and handle all process without prioritization. In the round-robin, each process gets a fixed time interval of the slice to utilize the resources or execute it's task called time quantum or time slice. Some of the round-robin processes are pre-empted if it executed in a given time slot, while the rest of the processes go back to the ready queue and wait to run in a circular order with the scheduled time slot until they complete their task. It removes the starvation for each process to achieve CPU shtedulling by proper partitioning of the CPU.

Algorithm :- i Organize all process according to their arrival time in the ready queue. The queue structure of ready queue is based on the FIFO structure to execute all CPU process.

ii Now, we push the first process from the ready queue to execute its task for a fixed time, allocated by each process that arrives in the queue.

iii If the process can't complete their task within defined time interval or slots because it is stopped by another process that pushes which helps ready queue to execute their task due to arrival time of the next process is reached; therefore, CPU saved the previous state of the process, which helps to resume from the point where it is interrupted.

iv Similarly, the scheduller selects another process from the ready queue to execute it's tasks. When a process finishes it's task within time slot, the process will not go for further execution.

v Repeat all the steps to execute remaining process.

Program :-

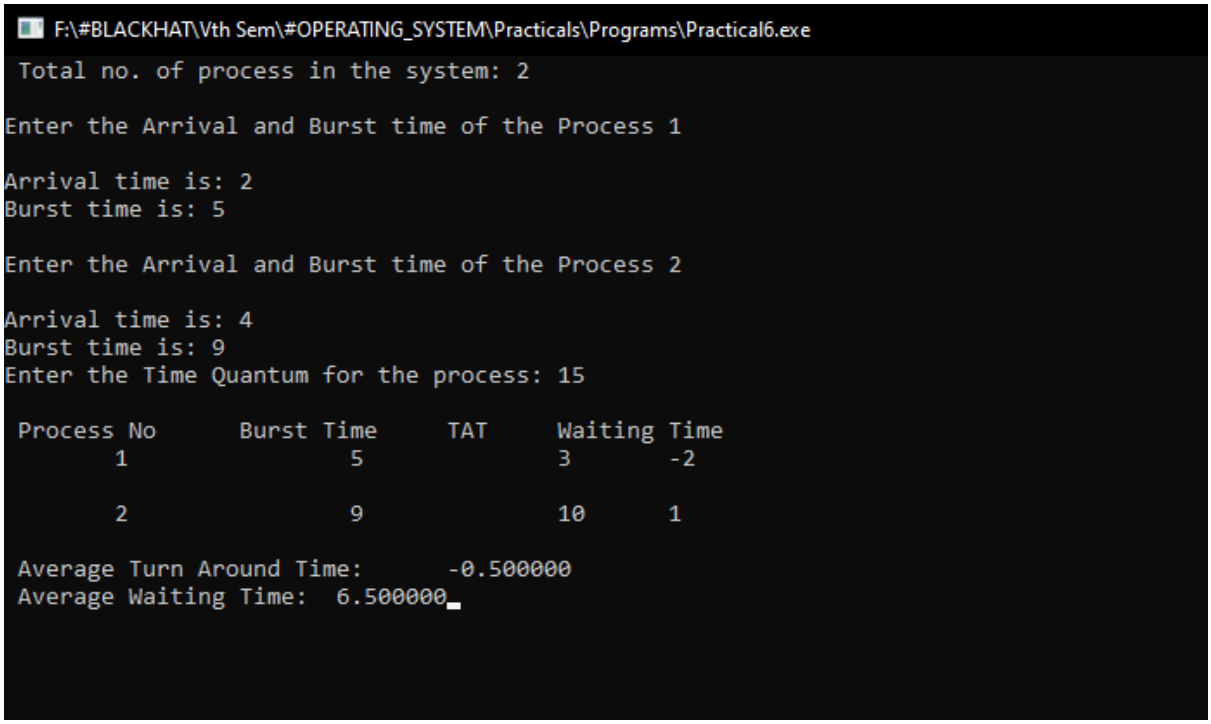
```
#include <stdio.h>
#include <conio.h>
void main () {
    int i, Nop, sum=0, count=0, y, quant,
        wt=0, tat=0, at[10], bt[10], temp[10];
    float avg_wt, avg_tat;
    printf("Total number of system in the
            process: ");
    scanf("%d", &Nop);
    y = Nop;
    for (i=0; i<Nop; i++) {
        printf("Enter arrival time and burst time
            of the process [%d]", i+1);
        printf("Arrival Time : ");
        scanf("%d", &at[i]);
```

```
        printf(" Burst Time: ");
        scanf("%d", &bt[i]);
        temp[i] = bt[i];
    }
printf("Enter time quantum : ");
scanf("%d", &quant);
printf("Process No. \t Burst Time \t TAT \t
            waiting Time \n");
for(sum=0; i=0; y!=0){
    if(sum temp[i] <= quant && tem[0]>0){
            sum += temp[i];
            temp[i] = 0;
            count = 1;
    }
    else if( temp[i]>0){
            temp[i] -= quant;
            sum += quant;
    }
    if( temp[i]==0 && count==1){
        y--;
        printf("Process No[%d] \t %d \t %d
                \t %d", i+1, bt[i], sum-at[
                    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
        f = 0;
avg_wt = wt * 1.0/Nop;
avg_tat = tat * 1.0/Nop;
printf(" Average Turn around time : %f", avg_tat);
printf(" Average waiting time : %f", avg_wt);

getch();
}
```

Conclusion :- Hence, we have successfully developed and executed a c program for CPU scheduling to simulate ~~Robi~~ Round Robin algorithm to find turnaround time and waiting time.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical6.exe
Total no. of process in the system: 2
Enter the Arrival and Burst time of the Process 1
Arrival time is: 2
Burst time is: 5
Enter the Arrival and Burst time of the Process 2
Arrival time is: 4
Burst time is: 9
Enter the Time Quantum for the process: 15
Process No Burst Time TAT Waiting Time
1 5 3 -2
2 9 10 1
Average Turn Around Time: -0.500000
Average Waiting Time: 6.500000

Practical No. 7

Aim:- Develop, debug and execute a C program to simulate priority CPU scheduling algorithm to find turnaround time and waiting time.

Theory :- Priority Scheduling is a time of a arrival of a process in the ready queue. It's priority is compared with the priority of the others processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The one with the highest priority among all the available processes will be given the CPU next.

The difference between premitive and non-premitive priority scheduling is that, the job which is being executed can be stopped at the arrival of a higher priority job.

Once all the jobs are available in a ready queue, the algorithm will behave as non-premitive scheduling, which means the job scheduled will run till the completion & no premition is done.

Algorithm :- i Make a structure process with variable pid, bt priority.

ii function compare(process a, process b)
return (a.priority > b.priority)

iii function waitingtime(process pro[], int n, int wt[])
set wt[0]=0
For loop i=1; i<n; i++
set wt[i] = pro[i-1].bt + wt[i-1]
end

iv function turnaround(process pro[], int n, int wt[], int tat[])
For loop i=0; i<n; i++
set tat[i] = pro[i].bt + wt[i]
end loop

v function avgtime(process pro[], int n)
declare wt[n], tat[n], total_wt=0, total_tat=0;
call waitingtime(pro, n, wt)
turnaround(pro, n, wt, tat)
print "process || Burst Time || waiting Time || Turnaround Time"
For loop i=0; i<n; i++
set total_wt = total_wt + wt[i]
total_tat = total_tat + tat[i]
end loop
print "process, burst time, waiting time, turnaround time"
print "Average time, Average Turnaround Time".

vi function Scheduling(process pro[], int n)
call sort(pro, pro+n, compare)
For loop i=0; i<n; i++
print order
end loop
call avgtime(pro, n)

VII function : main()

declare process pro[] = {{1, 10, 2}, {2, 5, 0}, {3, 8, 1}}

n = sizeof(pro[0])

call Scheduling (pro, n)

Stop.

Program :-

```
#include <stdio.h>
#include <conio.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 no. of process");
    scanf("%d", &n);
    printf(" Enter Burst time and priority :");
    for (i=0; i<n; i++) {
        printf(" P[%d]", i+1);
        printf(" Burst Time and Priority :");
        scanf(" %d, %d", &bt[i], &pr[i]);
        p[i] = i+1;
    }
    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];
        pr[i] = pr[pos];
        bt[pos] = temp;

        temp = p[i];
        p[i] = p[pos];
        p[pos] = temp;
    }
    wt[0] = 0;
    for(i=1; i<n; i++){
        wt[i] = 0;
        for(j=0; j<i; j++){
            wt[i] += bt[i];
        total += wt[i];
    }
    avg_wt = total / n;
    total = 0;
    printf("Process \t Burst Time \t Waiting Time \t
                                Turnaround Time");
    for(i=0; i<n; i++){
        tat[i] = bt[i] + wt[i];
        total += tat[i];
        printf("p[%d] \t %d \t %d \t %d", p[i], bt[i],
                                wt[i], tat[i]);
    }
    avg_tat = total / n;
    printf("Avg. W.T. & Avg. Turnaround Time : %d & %d",
                                avg_wt, avg_tat);

    return 0;
}
```

Conclusion :- Hence, we successfully developed and executed priority scheduling program.

```
F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical7.exe
Enter Burst Time and Priority

P[1]
Burst Time:8
Priority:2

P[2]
Burst Time:4
Priority:1

P[3]
Burst Time:6
Priority:3

P[4]
Burst Time:9
Priority:4

Process     Burst Time          Waiting Time    Turnaround Time
P[2]              4                 0                 4
P[1]              8                 4                 12
P[3]              6                 12                18
P[4]              9                 18                27

Average Waiting Time=8
Average Turnaround Time=15
```

Practical No. 8

Aim:- Develop, debug and execute a C program to simulate producer consumer problem using semaphores.

Theory :- The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enter them into the buffer. The consumer removes the items from the buffer and consumes them.

A producer should not produce items into the buffer when the consume is consuming an item from an buffer and vise-versa. So the buffer should only be accessed by the producer or consumer at a time. The producer consumer problems can be accessed or resolved using semaphores. The code for the producer and consumer process are as given follows:

Producer Process :-

```
do {
    produce items
    wait (empty);
    wait (empty);
    put them in Buffer
    signal (mutex);
    signal (Full);
} while (1);
```

Consumer Process :

```
do{
    wait (full);
    wait (mutex);
    Remove Item Buffer
    signal (mutex);
    signal (empty);
    consume Item
} while (1);
```

Program :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int mutex = 1, full=0, empty = 3, x=0;
int main () {
    int n;
    void producer ();
    void consumer();
    int wait (int);
    int signal(int);
    printf("1.Producer \n 2. Consumer \n 3. Exit ");
    while (1) {
        printf("\n Enter your choice: ");
        scanf("%d", &n);
        switch (n) {
            case 1 : if ((mutex==1) && (empty!=0))
                        producer ();
                     else
                        printf(" Buffer is full");
                     break;
```

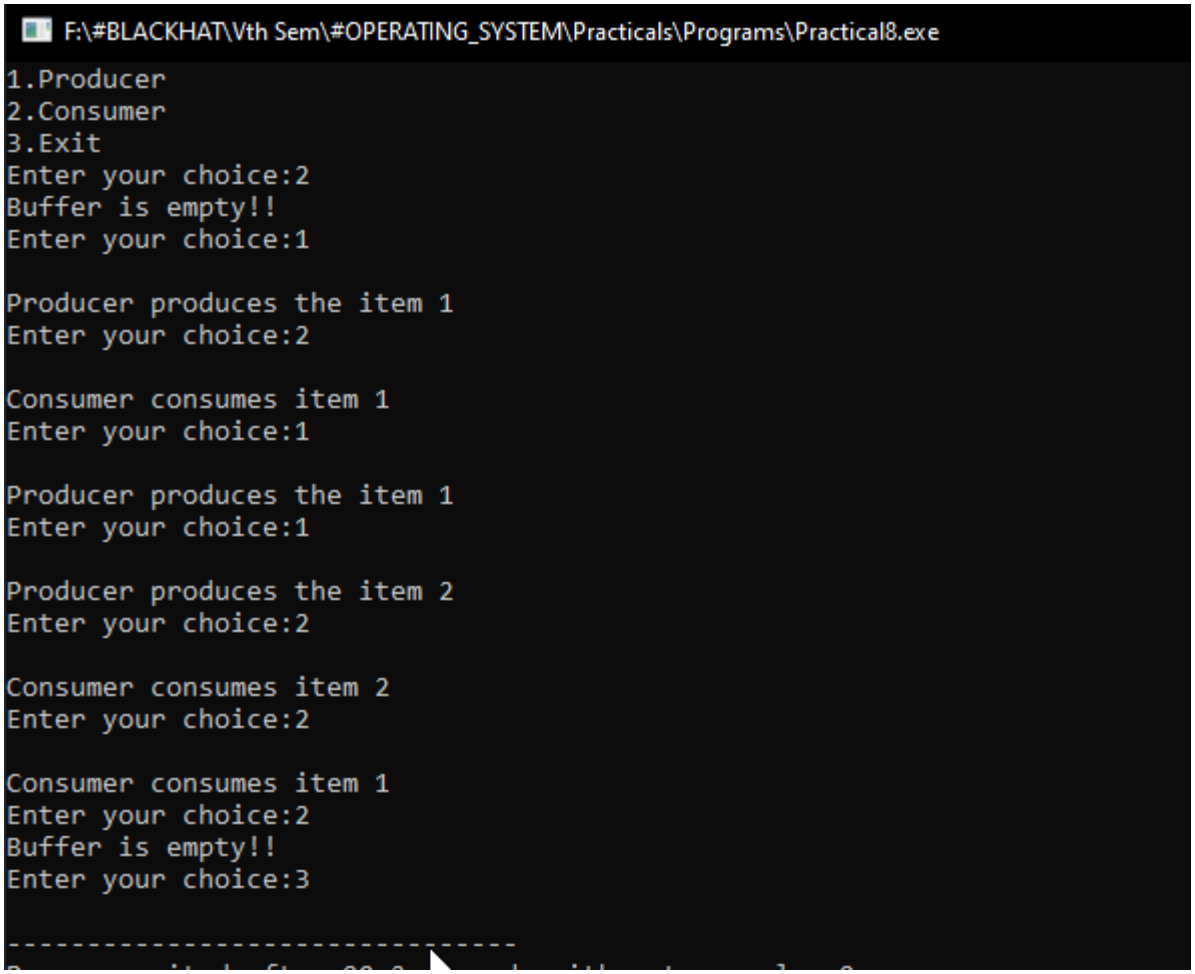
```
        case 2: if((mutex==1) && (full!=0))
                    consumer();
                else
                    printf("Buffer is Empty");
                break;
        case 3: exit(0);
                break;
        }
    }
    return 0;
}

int wait(int s){
    return (--s);
}
int signal(int s){
    return(++s);
}
void producer(){
    mutex = wait(mutex);
    Full = signal(Full);
    empty = wait(empty);
    x++;
    printf("producer produces the itemn %d", x);
    mutex = signal(mutex);
}

void consumer(){
    mutex = wait(mutex);
    Full = wait(full);
```

```
    Empty = signal(empty);
    printF(" Consumer consumes item %d", x);
    x--;
    mutex = signal(mutex);
}
```

Conclusion :- Hence, we have successfully developed and executed the consumer-producer problem using semaphores in C programming.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical8.exe
1.Producer
2.Consumer
3.Exit
Enter your choice:2
Buffer is empty!!
Enter your choice:1
Producer produces the item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:1
Producer produces the item 1
Enter your choice:1
Producer produces the item 2
Enter your choice:2
Consumer consumes item 2
Enter your choice:2
Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:3

Practical No. 9

Aim :- Develop, debug and executed C program to simulate FIFO page replacement algorithm.

Theory :- This is the simplest page replacement algorithm. operating system keeps tracks of all pages in the memory in a queue, oldest pages is in the front of the queue. When a page needs to be replaced page is the front of the queue is selected for removal.

~~Algorithm :- i Start transversing the pages.~~
~~ii Simultaneously maintain the pages in the queue to perform FIFO.~~
~~iii Increase page fault.~~

Algorithm :- i. Start traversing the pages.
IF set holds less pages, insert them into set ~~by~~ one by one until the size of set reaches it's capacity.
Else IF current pages is present in set, do nothing
Else remove first page from queue, and replace it in the queue with the current page in the string.

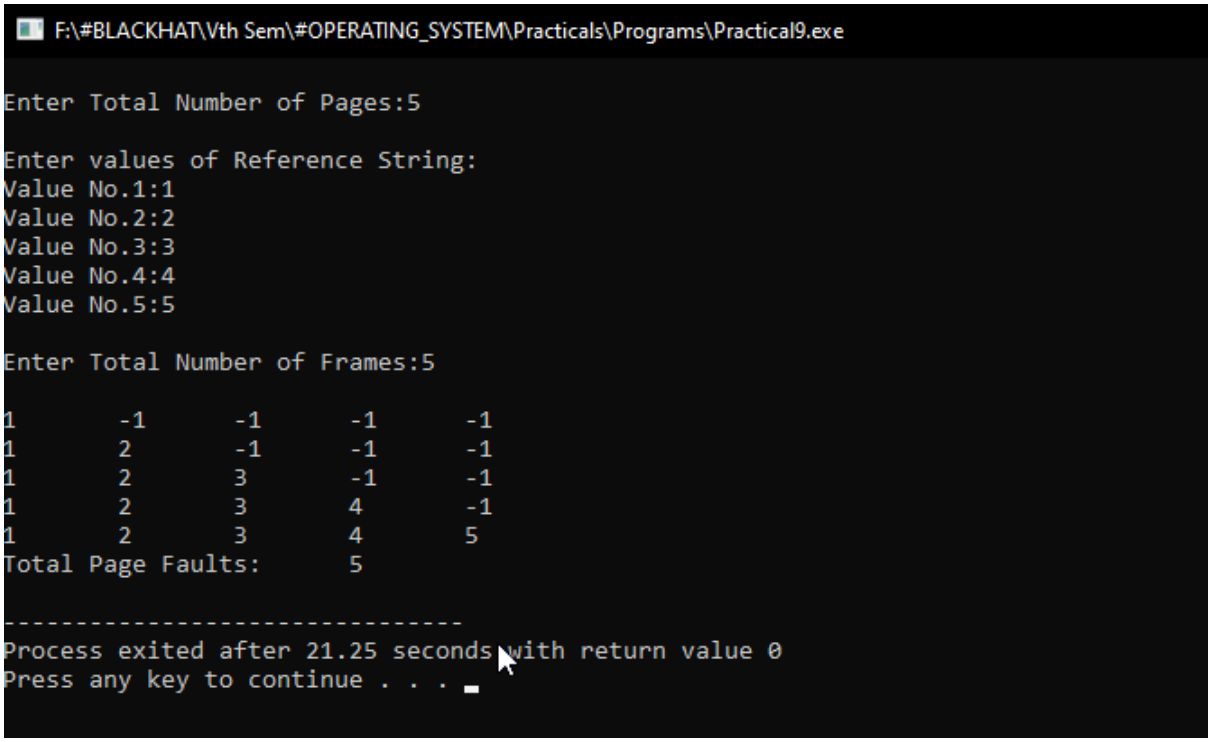
ii Store current page in queue
iii Increase page faults and return page faults.
iv END.

Program :-

```
#include <stdio.h>
int main() {
    int reference_string[10], page_faults=0, m, n, s, pages,
        frames;
    printf("Enter Total No. of Pages : ");
    scanf("%d", &pages);
    printf("Enter values of reference string : ");
    for(m=0; m<pages; m++) {
        printf("Value No. [%d] : ", m+1);
        scanf("%d", &reference_string[m]);
    }
    printf("Enter Total No. of frames : ");
    scanf("%d", &frames);
    int temp[frames];
    for(m=0; m<frames; m++)
        temp[m] = -1;
    for(m=0; m<pages; m++) {
        s = 0;
        for(n=0; n<frames; n++) {
            if(reference_string[m] == temp[n]) {
                s++;
                page_faults--;
            }
        }
        page_faults++;
        if((page_faults<=frames) && (s==0))
            temp[m] = reference_string[m];
        else if(s==0)
            temp[(page_faults-1) % frames] = reference_string[m];
```

```
    printF("\n");
    for (n=0 ; n< frames ; n++)
        printf("%d ", temp(n));
  }
  printf("Total page faults : %d \n", page_faults);
  return 0;
}
```

Conclusion :- Hence, we have successfully developed, and executed FIFO page replacement algorithm using C programming.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical9.exe
Enter Total Number of Pages:5
Enter values of Reference String:
Value No.1:1
Value No.2:2
Value No.3:3
Value No.4:4
Value No.5:5
Enter Total Number of Frames:5
1 -1 -1 -1 -1
1 2 -1 -1 -1
1 2 3 -1 -1
1 2 3 4 -1
1 2 3 4 5
Total Page Faults: 5
Process exited after 21.25 seconds with return value 0
Press any key to continue . . .

Practical No. 10

Aim :- Develop, debug and execute a C program to simulate LRU page replacement algorithm.

Theory :- The page replacement algorithm helps in deciding the memory pages that needs to be snapped out, written to the disk when a page of memory needs to be allocated in the system.

The LRU page replacement method is a marking algorithm. It keeps a track of the pages usage in a given period of time. It offers optimum performance but it is costly in it's implementation. It is modified for implementation and it's succesor are LRU-K and ARC algorithm.

Algorithm :- If i sets holds less pages than capacity,
insert into set one by one and
maintain index of each page in
map called indexes
ii Increase page fault.

Else if i Find the page in set which was recently
used and replace it with having minimum index
page.
ii Replace found page with current page.
iii Increment in page faults
iv Update index of current page and return
page faults.

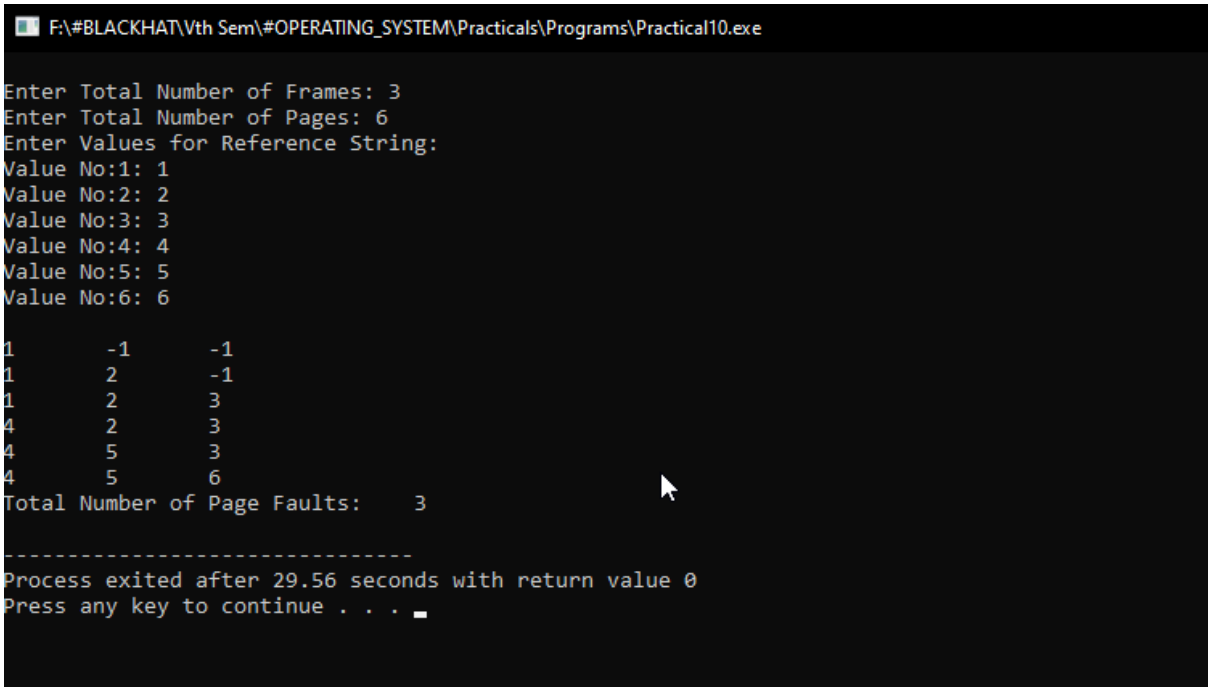
END

Program :-

```
#include <stdio.h>
int main () {
    int frames[10], temp[10], pages[10], total_pages, m, n,
        position, k, l, total_frames;
    int a=0, b=0, page_fault = 0;
    printf(" Enter Total No. of Frames ");
    scanf(" %d", &total_frames);
    for( m=0; m<total_frames ; m++)
        frames[m]= -1;
    printf(" Enter Total No. of Pages : ");
    scanf(" %d", &total_pages);
    printf(" Enter value for reference string: ");
    for ( m= 0 ; m< total_pages ; m++) {
        printf(" Value of Page No. [%d] : ", m+1);
        scanf("%d", &pages[m]); }
    for( n=0 ; n< total_pages ; n++) {
        a=b , b= 0;
        for( m=0; m< total_frames ; m++) {
            if( frames[m] == pages[n]) {
                a= 1;
                b= 1;
                break ; }
        }
        if (a==0) {
            for (m=0 ; m< total_frames ; m++){
                if ( frames[m] == -1) {
                    frames[m]= pages[n];
                    b=1;
                    break; }
            }
        }
```

```
if(b==0){
    for(m=0; m<total_frames; m++){
        temp[m]=0;
    for(k=n-1, l=1; l<=total_frames-1; l++, k--){
        for(m=0; m<total_frames; m++){
            if(frames[m]==pages[k]){
                temp[m]=1; }
        }
    }
for(m=0; m<total_frames; m++){
    if(temp[m]==0){
        position=m;
    frames[position]=pages[n];
    pages_fault++;
}
printf("\n")
for(m=0; m<total_frames; m++){
    printf("%d", frames[m]);
}
printf("Total No. of Pages faults: %d", pages_faults);
}
```

Conclusion :- Hence, we have successfully developed and executed LRU page replacement algorithm.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical10.exe
Enter Total Number of Frames: 3
Enter Total Number of Pages: 6
Enter Values for Reference String:
Value No:1: 1
Value No:2: 2
Value No:3: 3
Value No:4: 4
Value No:5: 5
Value No:6: 6
1 -1 -1
1 2 -1
1 2 3
4 2 3
4 5 3
4 5 6
Total Number of Page Faults: 3
Process exited after 29.56 seconds with return value 0
Press any key to continue . . .

Practical No. 11

Aim:- Develop and execute a C program to simulate optimal replacement algorithm.

Theory:- The page replacement algorithms are used in operating systems that use virtual memory mangt. When a page of memory needs to be allocated to the CPU, these page replacement algorithm decide which pages should be written to the disk and which should swapped out of memory. The algorithm is also know as Clairvoyant Replacement Algorithm.

When a page needs to be swapped into the memory, the OS will help swap out the page which is not required to be used in the near future. This page replacement algorithm is a little unrealiable to implement and therefore it cannot be implemented in a general purpose operating system.

Algorithm :- i If referred page is already present, increment in count.

ii If not present, if a page that is never referenced in future. If such a page exists, replace this page with new page.

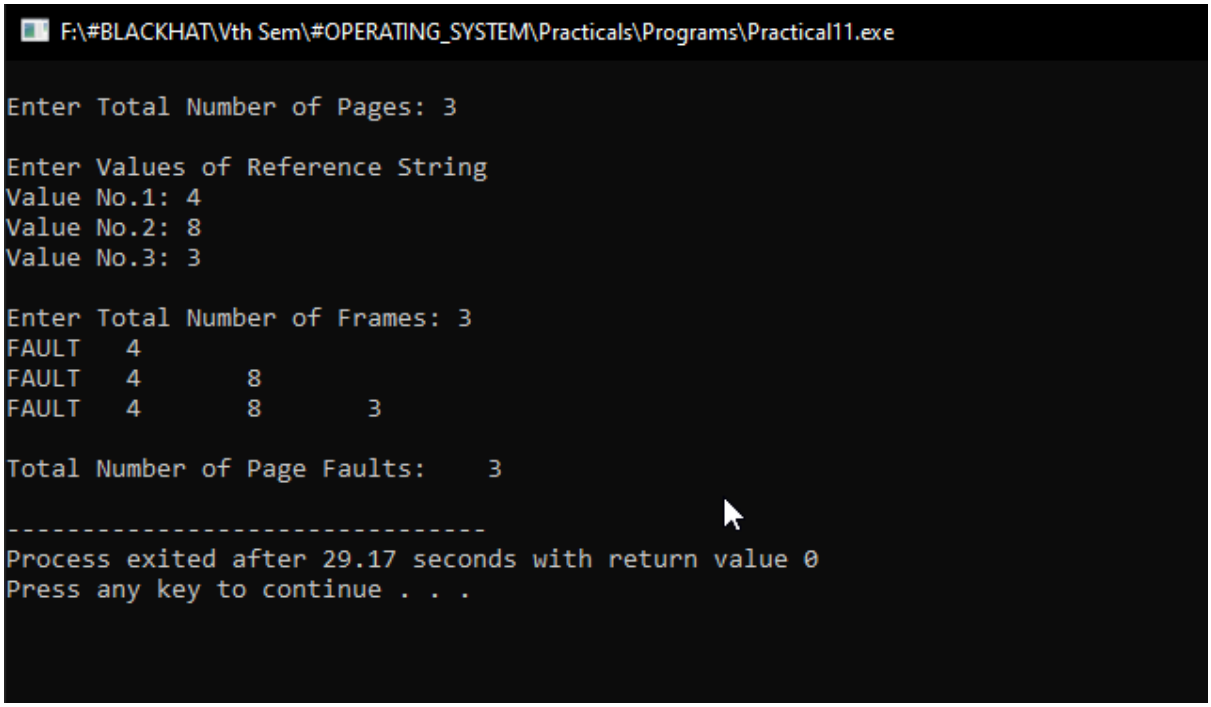
iii If no such page exists, find a page that is referenced Fartherest in future and replace it with new page.

Program :-

```
#include <stdio.h>
int main(){
    int reference_string[25], frames[25], interval
    int pages, total_frames, page_faults = 0;
    int m, n, temp, flag, found;
    int position, maximum_interval, previous_frame = -1;
    printf("Enter Total No. of pages ");
    scanf("%d", &pages);
    printf(" Enter values of reference string");
    for (m=0; m<pages; m++)
        scanf("%d", &reference_string[m]);
    printf(" Enter Total No. Frames: ");
    scanf(" %d ", &total_frames);
    for (m=0; m<total_frames; m++)
        frames[m] = -1;
    for (m=0; m< pages; m++){
        flag 0
        for (n=0; n<total_frames; n++){
            if( frame[n] == reference_string[m]){
            flag = 1;
            printf("\t");
            break; }
        }
    if (flag == 0){
        if( previous_frame. == total_frame-1){
            for (n=0, n< total_frame; n++){
                for (temp = m+1; temp < pages; temp++){
                    interval [n] = 0;
                    if( frames [n] == reference_string [temp]){
                        interval [n] = temp - m;
                        break; }
                }
            }
        }
```

```
        found = 0;
        for (n=0; n< total_frames; n++){
            if (interval [n] == 0){
                position = n;
                found = 1;
                break; }
            }
        }
        else{ position = ++ previous_frame;
            found = 1; }
        If (found == 0){
            maximum_interval = interval [0];
            position = 0;
            For( n=1; n < total_frames; n++){
                if(maximum_interval < interval [n]){
                    maximum_interval = interval[n];
                    position = n;
                }
            }
        }
        frames [position] = reference_string [m];
        printf(" FAULT");
        page_fault ++;
    }
    for (n=0; n< total_frames; n++){
        IF (frames [n] != -1)
            printf("%d", frames [n]);
        printf("\n");
    }
    printf(" Total No. of page fault : %d", page_fault);
}
```


Conclusion :- Hence, we have successfully developed and executed optimal page replacement algorithm using c program.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical11.exe
Enter Total Number of Pages: 3
Enter Values of Reference String
Value No.1: 4
Value No.2: 8
Value No.3: 3
Enter Total Number of Frames: 3
FAULT 4
FAULT 4 8
FAULT 4 8 3
Total Number of Page Faults: 3
Process exited after 29.17 seconds with return value 0
Press any key to continue . . .

Practical No. 12

Aim :- Develop, debug and execute C program to simulate LFU page replacement algorithm.

Theory :- In LFU page replacement method, the page with minimum count is selected for replacement with the page that needs to enter into the system. LFU is a cache algorithm which is used to manage computer memory. A counter is assigned to every block of memory that is loaded into the cache memory.

LFU algorithm sometimes also combined with LRU algorithm and them implemented.

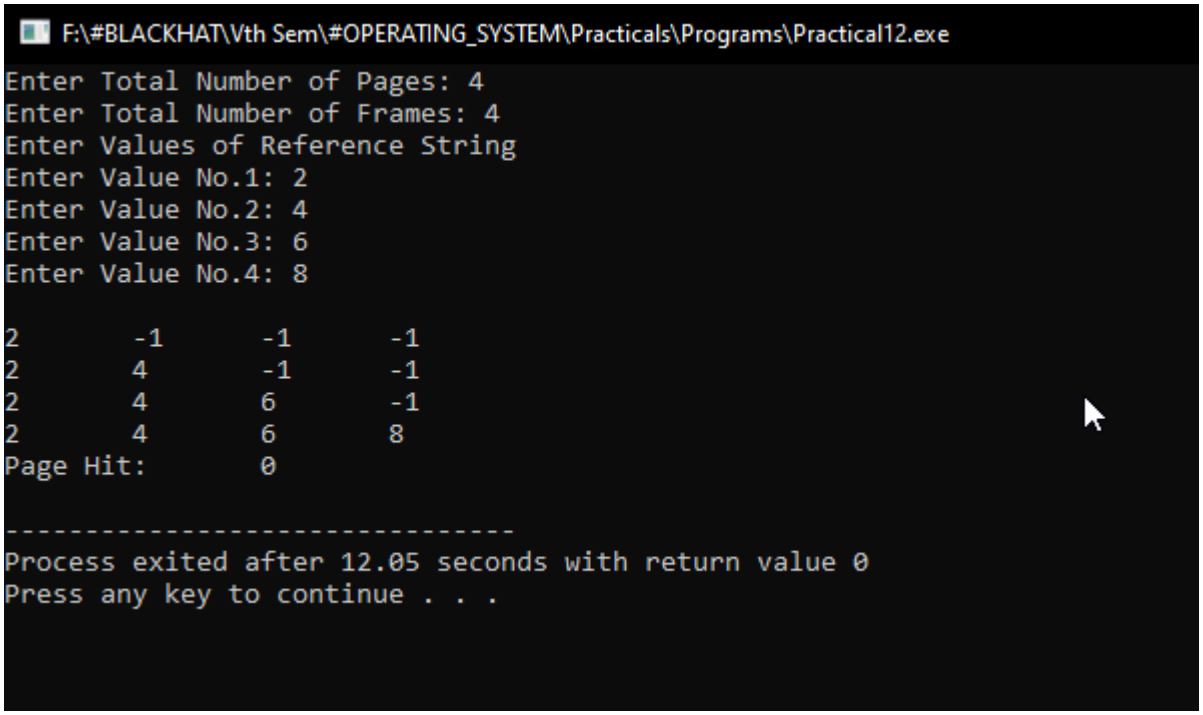
Program :-

```
#include <stdio.h>
int main (){
    int total_frames, total_pages, hit=0;
    int m, n, page, flag, k, minimum_time, temp;
    int page[20], fram[20], arr[20], time[20];
    printf("Enter total No. of Pages : ");
    scanf(" %d", &total_pages);
    printf(" Enter total No. of Frames : ");
    scanf(" %d", &total_frames);
    for (m=0; m<total_frames; m++
        frames[m] = -1;
    for (m=0; m<20; m++)
        arr[m] = 0;
    printf(" Enter value of reference string");
```

```
for(m=0; m<total_pages; m++) {
    printf("Value No. %d", 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 = fram[0];
    for(n=0; n<total_frames; n++) {
        if(frame[n]==-1 || fram[n]==pages[m]) {
            if(frame[n] != -1)
                hit++;
            Flag = 0;
            frame[n] = page[m];
            break;
        }
        if(arr[k] > arr[frame[m]])
            k = frame[n];
    }
    if(Flag) {
        minimum_time = 20;
        for(n=0; n<total_frames; n++) {
            if(arr[frame[n]] == arr[k] && time[fram[n]] <
                                        minimum_time) {
                temp = n;
                minimum_time = time[frame[n]];
            }
            arr[frame[temp]] = 0;
            fram[temp] = page[m];
        }
```

```
        for(n=0; n<totalFrames; n++);
            printf("%d", frames[n]);
        printf("\n");
    }
    printf("Page hit : %d", hit);
    return 0;
}
```

Conclusion :- Hence, we have successfully developed and executed C program to simulate LFU page replacement algorithm.


F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical12.exe
Enter Total Number of Pages: 4
Enter Total Number of Frames: 4
Enter Values of Reference String
Enter Value No.1: 2
Enter Value No.2: 4
Enter Value No.3: 6
Enter Value No.4: 8
2 -1 -1 -1
2 4 -1 -1
2 4 6 -1
2 4 6 8
Page Hit: 0
Process exited after 12.05 seconds with return value 0
Press any key to continue . . .

Practical. No. 13

Aim :- Develop, debug and execute a 'C' program to simulate the following contiguous memory allocation techniques: a) Worst-Fit
b) Best-Fit
c) First-Fit

Theory :- The main memory must accommodate both the operating system and the various user processes, therefore we need to allocate main memory in the most efficient way possible. This section explains once early method, contigerous memory allocation

a) Best-Fit - Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.

b) First-Fit :- Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-Fit search ended. We can stop searching as soon as we find a free hole that is largeg enough.

c) Worst-Fit - Allocate the largest hole, again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

Program :-

a) Best - fit

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main(){
   int frag[max], b[max], f[max], i, j, nb, nf, temp,
       lowest = 10000;
   static int bf[max], ff[max];
   printf("Enter the No. of blocks");
   scanf("%d", &nb);
   printf("Enter the number of files:");
   scanf("%d", &nf);
   printf("Enter the size of the blocks:");
   for( i=1; i<=nb; i++){
      printf("Block %d: ", i);
      scanf("%d", &b[i]);
   }
   printf("Enter the size of the files:");
   for(i=1; i<=nf; i++){
      printf("File %d", i);
      scanf("%d", &f[i]);
   }
   for( i=1; i<=nf; i++) {
      for( j=1; j<=nb; j++){
         if( bf[j] != 1) {
            temp = b[j] - f[i];
            if(temp>=0) {
               if ( lowest > temp) {
                  ff[i] = j;
                  lowest = temp;}
            }  }
```

```
        frag[i] = lowest;
        bf[ff[i]] = 1;
        lowest = 10000;
    }
    printf("File No. \t File Size \t Block No. \t
            Block Size \t fragment");
    for(i=0; i<nf && ff[i] != 0; i++){
        printf("%d \t %d \t %d \t %d \t %d", i, f[i],
                ff[i], b[ff[i]], flag[i]);
    }
}
```

b) First-fit

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main(){
    int frag[max], b[max], f[max], i, j, nb, nf, temp;
    static int bf[max], ff[max];
    printf("Enter the No. of blocks: ");
    scanf("%d", &nb);
    printf("Enter the no. of files");
    scanf("%d", &nf);
    printf("Enter the size of blocks");
    for(i=1; i<=nb; i++){
        printf("Block No: %d", i);
        scanf("%d", &b[i]);
    }
```

```
printf("Enter the size of files:");
for (i=1; i<=nf; i++){
    printf("File %d:", i);
    scanf("%d", &f[i]);
}
for (i=1; i<=nf; i++){
    for (j=1; j<=nb; j++){
        if (bf[j]!=1){
            temp = b[j] - f[i];
            if(temp>=0){
                ff[i]=j;
                break;}
        }
    }
frag[i] = temp;
    bf[ff[i]]=1;
}
printf("File No:\t File_Size:\t Block No:\t
                Block_size:\t Fragement");
for (i=1; i<=nf; i++)
    printf("%d\t%d\t%d\t%d\t%d", i, f[i],
                ff[i], b[ff[i]], frag[i]);
}
```

c) Worst-Fit

```
#include <stdio.h>
#include <conio.h>
#define maxe 25
void main() {
    int frag[maxe], b[maxe], f[maxe], i, j, nb, nf, temp,
        highest = 0;
    static int bf[maxe], ff[maxe];
    printf("Enter the number of blocks");
    scanf("%d", &nb);
    printf("Enter the No. of files");
    scanf("%d", &nf);
    printf("Enter the size of blocks");
    for(i=1; i<=nb; i++) {
        printf("Block. NO. %d:", i);
        scanf("%d", &b[i]);
    }
    printf("Enter the size of files");
    for(i=1; i<=nf; i++){
        printf("File NO. %d:", i);
        scanf("%d", &f[i]);
    }
    for(i=1; i<=nf; i++) {
        for(j=1; j<=nb; j++){
            if(bf[i]!=1) {
                temp = b[j]-f[i];
                if(temp>=0) {
                    if(highest<temp) {
                        ff[i]=j;
                        highest = temp;}
                }
            }
    }
```

```
        Frag [i] = highest;
        bF [ff[i]] = 1;
        highest = 0;
    }
    printf (" File No. \t File Size : \t Block No. \t
                Block Size \t Fragment ");
    for (i=1; i<=nf; i++)
        printf(" %d \t %d \t %d \t %d \t %d", i,
                f[i], ff[i], b[ff[i]], Frag[i]
    }
```

Conclusion :- Hence, we have successfully developed and executed c program to ~~conti~~ simulate the Best-Fit, First-Fit and worst-fit contigues memory allocation techniques.

```
F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical13_1.exe

               Best Fit


Enter the number of blocks:2
Enter the number of files:1

Enter the size of the blocks:-
Block 1:5
Block 2:2
Enter the size of the files :-
File 1:4

File No  File Size      Block No        Block Size      Fragment
   1           4               1               5               1
--------------------------------
Process exited after 18.59 seconds with return value 0
Press any key to continue . . .
```

```
F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical13_1.exe

               First Fit


Enter the number of blocks:4
Enter the number of files:3

Enter the size of the blocks:-
Block 1:2
Block 2:1
Block 3:5
Block 4:6
Enter the size of the files :-
File 1:2
File 2:3
File 3:5

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               2               1               2               0
2               3               3               5               2
3               5               4               6               1
--------------------------------
Process exited after 46.51 seconds with return value 0
Press any key to continue . . .
```

```
F:\#BLACKHAT\Vth Sem\#OPERATING_SYSTEM\Practicals\Programs\Practical13_3.exe

               Worst Fit


Enter the number of blocks:4
Enter the number of files:3

Enter the size of the blocks:-
Block 1:5
Block 2:7
Block 3:2
Block 4:6
Enter the size of the files :-
File 1:3
File 2:2
File 3:4

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               3               2               7               4
2               2               4               6               4
3               4               1               5               1
--------------------------------
Process exited after 24.02 seconds with return value 0
Press any key to continue . . .
```