Near ITPB, Channasandra, Bengaluru – 560067
Affiliated to VTU, Belagavi
Approved by AICTE, New Delhi
Recognized by UGC under 2(f) & 12(B)
Accredited by NBA & NAAC

DEPARTMENTOF COMPUTER SCIENCE AND ENGINEERING

III SEMESTER

MVJ22CS32 – OPERATING SYSTEMS

ACADEMIC YEAR 2024– 2025(ODD)

LABORATORYMANUAL

NAM E OF THE STUDENT	**:**	
BRANCH	**:**	
UNIVERSITY SEAT NO.	**:**	
SEMESTER & SECTION	**:**	
BATCH	**:**	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To create an ambiance in excellence and provide innovative emerging programs in Computer Science and Engineering and to bring out future ready engineers equipped with technical expertise and strong ethical values.

MISSION:

1. Concepts of Computing Discipline: To educate students at undergraduate, postgraduate and doctoral levels in the fundamental and advanced concepts of computing discipline.

2. Quality Research: To provide strong theoretical and practical background across the Computer Science and Engineering discipline with the emphasis on computing technologies, quality research, consultancy, and trainings.

3. Continuous Teaching Learning: To promote a teaching learning process that brings advancements in Computer Science and Engineering discipline leading to new technologies and products.

4. Social Responsibility and Ethical Values: To inculcate professional behavior, innovative research Capabilities, leadership abilities and strong ethical values in the young minds so as to work with the commitment for the betterment of the society

PROGRAMEDUCATIONALOBJECTIVES(PEOs):

PEO1: Current Industry Practices: Graduates will analyze real world problems and give solution using current industry practices in computing technology.

PEO2: Research and Higher Studies: Graduates with strong foundation in mathematics and engineering fundamentals that will enable graduates to pursue higher learning, R&D activities and consultancy.

PEO3: Social Responsibility: Graduates will be professionals with ethics, who will provide industry growth and social transformation as responsible citizens.

PEO4: Entrepreneur: Graduates will be able to become entrepreneur to address social, technical and business challenges.

PROGRAMOUTCOMES(POs):

Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the Engineering practice.

Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design Documentation, make effective presentations, and give and receive clear instructions.

Project management and finance: Demonstrate knowledge and understanding of the engineering and Management principles and apply these to one's own work, as a member and leader in a team, to manage Projects and
In multidisciplinary environments.

Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMSPECIFICOUTCOMES(PSOs):

PSO1: Programming: Ability to understand, analyze and develop computer programs in theareas related to algorithms, system software, multimedia, web design, DBMS, and networkingforefficientdesign of computer-based systemsofvaryingcomplexity.

PSO2:PracticalSolution: Abilitytopracticallyprovidesolutionsforrealworldproblemswith a broad range of programming language and open-source platforms in various computingdomains.

PSO3: Research: Ability to use innovative ideas to do research in various domains to solvesocietalproblems.

COURSEOBJECTIVES:

- Foundation knowledge in database concepts, technology and practice to groom studentsintowell-informed databaseapplicationdevelopers.
- StrongpracticeinSQLprogramming through avariety ofdatabase problems.
- Develop database applications using front-end tools and back-end DBMS.

PREREQUISITES:

Basic programming Languages like C.

.

COURSEOUTCOMES(CO's):

At the end of the course, the student will be able to:

CO 1. Explain the structure and functionality of operating system

CO 2. Apply appropriate CPU scheduling algorithms for the given problem.

CO 3. Analyze the various techniques for process synchronization and deadlock handling.

CO 4. Apply the various techniques for memory management

CO 5. Explain file and secondary storage management strategies. CO 6. Describe the need for information protection mechanisms

CONTENTS

Sl.NO	Experiments
1	Develop a c program to implement the Process system calls (fork (), exec(), wait(), create process, terminate process)
2	Simulate the following CPU scheduling algorithms to find turnaround time and waiting time a) FCFS b) SJF c) Round Robin d) Priority.
3	Develop a C program to simulate producer-consumer problem using semaphores.
4	Develop a C program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.
5	Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.
6	Develop a C program to simulate the following contiguous memory allocation Techniques: a) Worst fitb) Best fitc) First fit.
7	Develop a C program to simulate page replacement algorithms: a) FIFO b) LRU
8	Simulate following File Organization Techniques a) Single level directoryb) Two level directory
9	Develop a C program to simulate the Linked file allocation strategies.
10	Develop a C program to simulate SCAN disk scheduling algorithm.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

1. IPCC means practical portion integrated with the theory of the course.
2. CIE marks for the theory component are **25 marks** and that for the practical component is **25 marks**.

25 marks for the theory component are split into **15 marks** for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and **10 marks** for other assessment methods

mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.

1. Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for **25 marks)**.
2. The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC.CIE **for the practical component of the IPCC**
3. **15 marks** for the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
4. On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
5. The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to **15 marks**.
6. The laboratory test **(duration 02/03 hours)** after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks.**
7. Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.

The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCCSEE **for IPCCTheory SEE will be conducted by University as per the scheduled timetable, with common question papers for** the course (**duration 03 hours**)[The question paper will have ten questions. Each question is set for 20 marksThere will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that moduleThe students have to answer 5 full questions, selecting one full question from each module]Marks scoredby the student shall be proportionally scaled down to 50 Marks]The theory **portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a]CIE component only. Questions mentioned in the SEE paper may include questions from the practical component**

	BASICS OF UNIX COMMANDS
	INTRODUCTION TO UNIX

AIM:

To study about the basics of UNIX

UNIX:

It is a multi-user operating system. Developed at AT& T Bell Industries, USA in 1969.

Ken Thomson along with Dennis Ritchie developed it from MULTICS (Multiplexed Information and Computing Service) OS.

By1980, UNIX had been completely rewritten using C language.

LINUX:

It is similar to UNIX, which is created by Linus Torualds. All UNIX commands works in Linux. Linux is a open source software. The main feature of Linux is coexisting with other OS such as windows and UNIX.

STRUCTURE OF A LINUXSYSTEM:

It consists of three parts.

UNIX kernel
Shells
Tools and Applications

UNIX KERNEL:

Kernel is the core of the UNIX OS. It controls all tasks, schedule all Processes and carries out all the functions of OS.

Decides when one programs tops and another starts.

SHELL:

Shell is the command interpreter in the UNIX OS. It accepts command from the user and analyses and interprets them

	BASICS OF UNIX COMMANDS
	BASIC UNIX COMMANDS

AIM:

To study of Basic UNIX Commands and various UNIX editors such as vi, ed, ex and EMACS.

CONTENT:

Note: Syn->Syntax

date

–used to check the date and time Syn:$date

Format	Purpose	Example	Result
+%m	To display only month	$date+%m	06
+%h	To display month name	$date+%h	June
+%d	To display day of month	$date+%d	O1
+%y	To display last two digits of years	$date+%y	09
+%H	To display hours	$date+%H	10
+%M	To display minutes	$date+%M	45
+%S	To display seconds	$date+%S	55

cal

–used to display the calendar Syn:$cal 2 2009

echo

–used to print the message on the screen.
Syn:$echo “text”

ls

–used to list the files. Your files are kept in a directory.
Syn:$lsls–s

All files (include files with prefix)
ls–l Lodetai (provide file statistics)
ls–t Order by creation time
ls– u Sort by access time (or show when last accessed together with –l) ls–s Order by size
ls–r Reverse order
ls–f Mark directories with /,executable with* , symbolic links with @, local sockets with =,
named pipes(FIFOs)with
ls–s Show file size
ls– h“ Human Readable”, show file size in Kilo Bytes & Mega Bytes (h can be used together with –l or)

ls[a-m]*List all the files whose name begin with alphabets From „a“ to „m“ ls[a]*List all the files whose name begins with „a“ or „A“
Eg:$ls>my list Output of „ls“ command is stored to disk file named „my list“

lp

–used to take printouts Syn:$lp filename

man

–used to provide manual help on every UNIX commands.
Syn:$man unix command
$man cat

who &whoami

–it displays data about all users who have logged into the system currently. The next command displays about current user only.
Syn:whowhoami

uptime

–tells you how long the computer has been running since its last reboot or power-off.
Syn:$uptime

uname

–it displays the system information such as hardware platform, system name and processor, OS type.
Syn:$uname–a

hostname

–displays and set system host name Syn:$ hostname

bc

–stands for „best calculator"

$bc	$ bc	$ bc	$ bc
10/2*3	scale =1	ibase=2	sqrt(196)
15	2.25+1	obase=16	14 quit
	3.35	11010011	
	quit	89275	
		1010	
		Ā	
		Quit	
$bc	$ bc-l		
for(i=1;i<3;i=i+1)I	scale=2		
1	s(3.14)		
2	0		
3 quit			

FILE MANIPULATION COMMANDS

cat–this create, view and concatenate files.

Creation:

Syn:$cat>filename

Viewing:

Syn:$cat filename

Add text to an existing file:

Syn:$cat>>filename

Concatenate:

Syn:$catfile1file2>file3
$catfile1file2>>file3 (no over writing of file3)

grep–used to search a particular word or pattern related to that word from the file. Syn:$grep search word filename
Eg:$grep anu student

rm–deletes a file from the file system Syn:$rm filename

touch–used to create a blank file.
Syn:$touch file names

cp–copies the files or directories Syn:$cpsource file destination file Eg:$cp student stud

mv–to rename the file or directory syn:$mv old file new file
Eg:$mv–i student student list(-i prompt when overwrite)

cut–it cuts or pickup a given number of character or fields of the file. Syn:$cut<option><filename>
Eg: $cut –c filename
$cut–c1-10emp
$cut–f 3,6emp
$ cut –f 3-6 emp
-c cutting columns
-f cutting fields

head–displays10 lines from the head(top)of a given file Syn:$head filename
Eg:$head student
To display the top two lines:

Syn:$head-2student

tail–displays last 10 lines of the file Syn:$tail filename
Eg:$tail student
To display the bottom two lines;
Syn:$ tail -2 student

chmod–used to change the permissions of a file or directory. Syn:$chmodcategoryoperationpermission file
Where, Category–is the user type
Operation–is used to assign or remove permission Permission–is the type of permission
File–are used to assign or remove permission all

Examples:
$chmodu-wx student
Removes write and execute permission for users
$chmodu+rw,g+rwstudent
Assigns read and write permission for users and groups
$chmodg=rwx student
Assigns absolute permission for groups of all read, write and execute permissions

wc–it counts the number of lines, words, character in a specified file(s) with the options as –l,-w,-c

Category	Operation	Permission

u– users g–group o–others	+assign -remove =assign absolutely	r– read w– write x-execute

Syn: $wc –l filename
$wc –w filename
$wc–c filename

	BASICS OF UNIX COMMANDS
	UNIX EDITORS

AIM:

To study of various UNIX editors such as vi, ed, ex and EMACS.

CONCEPT:

Editor is a program that allows user to see a portions a file on the screen and modify characters and lines by simply typing at the current position. UNIX supports variety of Editors.
They are:vi,ed,ex,EMACS

Vi- vi is stands for "visual".vi is the most important and powerful editor.vi is a full screen editor that allows user to view and edit entire document at the same time.vi editor was written in the University of California, at Berkley by Bill Joy, who is one of the co-founder of Sun Microsystems.

Features of vi:

It is easy to learn and has more powerful features.

Itworksgreatspeedandiscasesensitive.vihaspowerfulundofunctionsandhas3modes:

Command mode

Insert mode

Escape or ex mode

In command mode, no text is displayed on the screen.

In Insert mode, it permits user to edit insert or replace text. In escape mode, it displays commands at command line.

Moving the cursor with the help of h, l, k, j, I, etc

EMACS Editor

Motion Commands:

M-> Move to end of file

M-< Move to beginning of file

C-v Move forward a screen M –v Move backward a screen C –n Move to next line C-p Move to previous line

C-a Move to the beginning of the line C-e Move to the end of the line

C-f Move forward a character C-b Move backward a character M-f Move forward a word

M-b Move backward a word

Deletion Commands:

DELdelete the previous character C -d delete the current character M -DEL delete the previous word

M-ddelete the next word

C-x DELdeletes the previous sentence

M-kdelete the rest of the current sentence

C-kdeletes the rest of the current line

C-xuundo the lasted it change

Search and Replace in EMACS:

yChange the occurrence of the pattern

nDon‟tchange the occurrence, but look for the other q Don‟t change. Leave query replace completely

!Change this occurrence and all others in the file

1.Develop a c program to implement the Process system calls(fork(),exec(),wait(),createprocess, terminate process)

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

main(void) {
pid_tpid = 0;

pid = fork();
        if (pid == 0) {
printf("I am the child.\\n");
        }
        if (pid> 0) {
printf("I am the parent, the child is %d.\\n", pid);
        }
        if (pid< 0) {
perror("In fork():");
        }

exit(0);
}
```

2. Simulate the following CPU scheduling algorithms to find turnaround time and waiting time

a) FCFS
b) SJF

c) Round Robin

d)Priority.

```
#include<stdio.h>
#include<conio.h>
main()
{
intbt[20],wt[20],tat[20],i,n;floa
 twtavg,tatavg;
clrscr();
printf("\nEnter the number of processes --
                ");scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnterBurstTimeforProcess%d--
",i);scanf("%d",&bt[i]);
}
wt[0]=wtavg=0;tat[0] =
tatavg                =
bt[0];for(i=1;i<n;i++)
{
wt[i]=wt[i-1]+bt[i-1];
tat[i]=tat[i-
1]+bt[i];wtavg=wtavg+w
t[i];tatavg=tatavg+ tat[i];
}
printf("\tPROCESS \tBURSTTIME\tWAITINGTIME\tTURNAROUNDTIME\n");
for(i=0;i<n;i++)
```

```
        printf("\n\tP%d\t\t%d\t\t%d\t\t%d",i,bt[i],wt[i],tat[i]);printf("\nAverage
        Waiting Time--%f",wtavg/n);
    printf("\nAverageTurnaroundTime--
    %f",tatavg/n);getch();
    }
```

INPUT

```
Enterthenumberofprocesses--          3
EnterBurstTimeforProcess0--          24
EnterBurstTimeforProcess1 --         3
EnterBurstTimeforProcess2--          3
```

OUTPUT			
PROCESS	BURSTTIME	WAITINGTIME	TURNAROUND TIME
P0	24	0	24
P1	3	24	27
P2	3	27	30
AverageWaitingTime--	17.000000		
AverageTurnaroundTime--		27.000000	

SOURCE CODE :

```
#include<stdio.h>
#include<conio.h>
main()
{
intp[20],bt[20],wt[20],tat[20],i,k,n,temp;floatwtavg,tatavg;
clrscr();
printf("\nEnter the number of processes --
");scanf("%d",&n);
```

```
for(i=0;i<n;i++)
{
p[i]=i;
printf("Enter Burst Time for Process %d -- ",
i);scanf("%d",&bt[i]);

}
for(i=0;i<n;i++)for(
k=i+1;k<n;k++)if(b
t[i]>bt[k])
{
temp=bt[i];
bt[i]=bt[k];
bt[k]=temp;

temp=p[i];
p[i]=p[k];p
[k]=temp;
}
wt[0]=wtavg=0;
tat[0] =tatavg =bt[0];for(i=1;i<n;i++)
{
wt[i]=wt[i-1]+bt[i-1];
 tat[i]=tat[i-
1]+bt[i];wtavg = wtavg +
wt[i];tatavg      =tatavg+
tat[i];
}
printf("\n\tPROCESS\tBURSTTIME\tWAITINGTIME\tTURNAROUNDTIME\n");
for(i=0;i<n;i++)
```

```
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i],
        tat[i]);printf("\nAverageWaitingTime--%f",wtavg/n);

printf("\nAverageTurnaroundTime --%f", tatavg/n);

getch();

}
```

INPUT

```
Enterthenumberofprocesses--          4
EnterBurstTimeforProcess0--          6
EnterBurstTimeforProcess1--          8
EnterBurstTimeforProcess2--          7
EnterBurstTimeforProcess3--          3
```

OUTPUT

PROCESS	BURST TIME	WAITING TIME	TURNARO UNDTIME
P3	3	0	3
P0	6	3	9
P2	7	9	16
P1	8	16	24
AverageWaitingTime --		7.000000	
AverageTurnaroundTime--		13.000000	

SOURCECODE

```
#include<stdio.h>

main()

{

int
      i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
floatawt=0,att=0,temp=0;

clrscr();

printf("Enterthenoofprocesses--
");scanf("%d",&n);

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

{

printf("\nEnterBurstTimeforprocess%d--
",i+1);scanf("%d",&bu[i]);
```

```
ct[i]=bu[i];
}
printf("\nEnterthesizeoftimeslice--
");scanf("%d",&t);
max=bu[0];for(i=1;i<n;i
++)if(max<bu[i])max=b
u[i];for(j=0;j<(max/t)+1;
j++)for(i=0;i<n;i++)if(b
u[i]!=0)
 if(bu[i]<=t)
                    {ta
t[i]=temp+bu[i];te
mp=temp+bu[i];b
u[i]=0;
 }
 else{bu[i]=b
u[i]-
t;temp=temp+
t;
 }
for(i=0;i<n;i++){
wa[i]=tat[i]-
ct[i];att+=tat[i];awt
+=wa[i];}
printf("\nTheAverageTurnaroundtimeis--
%f",att/n);printf("\nTheAverageWaitingtimeis--%f",awt/n);
printf("\n\tPROCESS\tBURSTTIME\tWAITINGTIME\tTURNAROUNDTIME\n");
for(i=0;i<n;i++)
printf("\t%d\t%d\t\t%d\t\t%d\n",i+1,ct[i],wa[i],tat[i]);
getch();
}
```

INPUT:

Enterthenoofprocesses–3

Enter Burst Time for process 1 –
24EnterBurstTimeforprocess2--
3EnterBurstTimeforprocess3–
3Enterthesize oftimeslice–3

OUTPUT: PROCESS	BURSTTIME	WAITINGTIME	TURNAROUNDTIME
1	24	6	30
2	3	4	7
3	3	7	10

TheAverageTurnaroundtimeis–
15.666667TheAverageWaitingtimeis5.666667

SOURCECODE:

```
#include<stdio.h>
main()
{
intp[20],bt[20],pri[20],wt[20],tat[20],i,k,n,temp;floatwtavg,tatavg;
clrscr();
printf("Enterthenumberofprocesses---
");scanf("%d",&n);
for(i=0;i<n;i++){
p[i]= i;
printf("EntertheBurstTime&PriorityofProcess%d---",i);scanf("%d
%d",&bt[i],&pri[i]);
}
for(i=0;i<n;i++)for(
k=i+1;k<n;k++)if(p
ri[i]
>pri[k]){temp=p[i];
```

```
p[i]=p[k];p[k]=tem
p;temp=bt[i];bt[i]=
bt[k];bt[k]=temp;te
mp=pri[i];pri[i]=pri
[k];pri[k]=temp;
}
wtavg=wt[0]=0;tatavg
=          tat[0]          =
bt[0];for(i=1;i<n;i++)
{
wt[i]=wt[i-1] +bt[i-1];
tat[i] =tat[i-1]+bt[i];

wtavg=wtavg+wt[i];tata
 vg =tatavg+ tat[i];
}
printf("\nPROCESS\t\tPRIORITY\tBURSTTIME\tWAITINGTIME\tTURNAROUNDTIME"
);
for(i=0;i<n;i++)
printf("\n%d \t\t %d \t\t %d \t\t %d \t\t %d
",p[i],pri[i],bt[i],wt[i],tat[i]);printf("\nAverage Waiting Time is ---
%f",wtavg/n); printf("\nAverageTurnaroundTimeis---%f",tatavg/n);
getch();
}
```

INPUT

```
Enterthenumberofprocesses–5                    3
EntertheBurstTime&PriorityofProcess 0---10
EntertheBurstTime&PriorityofProcess 1---1                    1
EntertheBurstTime&PriorityofProcess 2---2                    4
EntertheBurstTime&PriorityofProcess 3---1                    5
EntertheBurstTime&PriorityofProcess 4---5                    2
```

OUTPUT

PROCESS 1	PRIORITY 1	BURSTTIME 1	WAITI NG TIME0	TURNARO UND TIME1
4	2	5	1	6
0	3	10	6	16
2	4	2	16	18
3	5	1	18	19

AverageWaitingTimeis--- 8.200000

AverageTurnaroundTimeis---------- 12.000000

3. Develop a C program to simulate producer-consumer problem using semaphores.

```
#include<stdio.h
>
voidmain()
{
    intbuffer[10],bufsize,in,out,produce,consume,choice=
    0;in=0;
    out=0;
    bufsize=10;
    while(choice!=3)
    {
        printf("\n1. Produce \t 2. Consume \t3.
                Exit");printf("\nEnteryourchoice:");
        scanf("%d",&choice);
        switch(choice){
            case1:if((in+1)%bufsize==out)
                        printf("\nBufferisFull");
                  else
                  {     printf("\nEnterthevalue:");s
                        canf("%d",&produce);buffe
                        r[in]= produce;
                        in=(in+1)%bufsize;
                  }
                  break;;;
              case2:  if(in==out)
                        printf("\nBufferisEmpty");
```

```
            }       }               }
    else
    {
    consume=buffer[out];
    printf("\nThe consumed value is %d", consume);
    out = (out+1)%bufsize;
    }}
    break;
```

OUTPUT

```
1.Produce    2.Consume    3.
             ExitEnteryourchoice:2
BufferisEmpty
1.Produce    2.Consume    3.
             ExitEnteryourchoice:1
Enterthevalue:100
1.Produce    2.Consume    3.
             ExitEnteryourchoice:2
Theconsumedvalueis100
1.Produce    2.Consume    3.
             ExitEnteryourchoice:3
```

4**.Develop a C program which demonstrates inter process communication between a reader process and a writer process. Use mkfifo,open, read,write and close APIs in your program.**

```
/*WriterProcess*/


#include <stdio.h>#include
<fcntl.h>#include
<sys/stat.h>#include
<sys/types.h>#include<unistd.h>


intmain()
{
  int fd;
  char buf[1024];


  /* create the FIFO (named pipe) */char * myfifo
  = "/tmp/myfifo";mkfifo(myfifo, 0666);
  printf("RunReaderprocesstoreadtheFIFOFile\n");fd=open(myfifo,
  O_WRONLY);
  write(fd,"Hi",sizeof("Hi"));



  /*write"Hi"totheFIFO*/close(fd);
  unlink(myfifo);    /* remove the FIFO */return 0;
}


/*ReaderProcess*/
```

```
#include <fcntl.h>#include
<sys/stat.h>#include
<sys/types.h>#include
<unistd.h>#include<stdio.h>

#define MAX_BUF 1024int main()

{

 int fd;



 /*AtempFIFOfileisnotcreatedin reader*/char *myfifo =
 "/tmp/myfifo";

 charbuf[MAX_BUF];




 /* open, read, and display the message from the FIFO
 */fd=open(myfifo, O_RDONLY);

 read(fd, buf,
 MAX_BUF);printf("Writer:%s\n",buf);
 close(fd);

 return 0;

}
```

5. Develop a C program to simulate Bankers Algorithm for Dead Lock Avoidance.

```
#include<stdio.h>
#include<conio.h>
voidmain()
{
charjob[10][10];
inttime[10],avail,tem[10],temp[10];intsafe[10];i
nt          ind=1,i,j,q,n,t;
clrscr();
printf("Enternoofjobs:");sca
nf("%d",&n);for(i=0;i<n;i+
+)
{
printf("Enter    name    and    time:
                    ");scanf("%s%d",&job[
i],&time[i]);
}
printf("Enter     the     available
                    resources:");scanf("%d",&avail
);
for(i=0;i<n;i++)
{
temp[i]=time[i];
tem[i]=i;
}
for(i=0;i<n;i++)fo
r(j=i+1;j<n;j++)
{
if(temp[i]>temp[j])
{
```

```
t=temp[i];
temp[i]=temp[j];
temp[j]=t;t=tem[i];
tem[i]=tem[j];
tem[j]=t;
}
 }
for(i=0;i<n;i++)
{
q=tem[i];if(time[
q]<=avail)
{
safe[ind]=tem[i];avail=ava
il-
tem[q];printf("%s",job[safe
[ind]]);ind++;
}
else
{
printf("Nosafesequence\n");
}
}
printf("Safesequenceis:");f
or(i=1;i<ind;i++)
printf("%s
%d\n",job[safe[i]],time[safe[i]]);getch();
}
```

OUTPUT:

```
Enter noofjobs:4

Enter name and time: A
1Enter name and time: B
4Enter name and time: C
2Enternameandtime:D3

Enter the available resources:
20Safesequenceis:A1,C2,D3,B4.
```

6. Develop a Cprogram to simulate the following contiguous memory allocation Techniques:
a) Worst fit b)Best fit c) Firstfit.

PROGRAM

WORST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max
25voidmain()
{
    intfrag[max],b[max],f[max],i,j,nb,nf
    ,temp;staticintbf[max],ff[max];clrsc
    r();
    printf("\n\tMemory Management Scheme - First
                Fit");printf("\nEnterthenumberofblocks:"
    );
    scanf("%d",&nb);
    printf("Enterthenumberoffiles:");scanf("
    %d",&nf);
    printf("\nEnter the size of the blocks:-
    \n");for(i=1;i<=nb;i++)
    {
        printf("Block%d:",i);
        scanf("%d",&b[i]);
    }
    printf("Enter the size of the files :-
    \n");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("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement");for
(i=1;i<=nf;i++)printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]]
,frag[i]);getch();
}
```

INPUT

Enterthenumberofblocks:3En
terthenumberoffiles: 2

Enterthesizeoftheblocks:-
Block1:5

Block2: 2

Block3: 7

Enterthesizeofthefiles:-File
1:1

File2:4

OUTPUT

FileNo	FileSize	BlockNo	BlockSize	Fragment
1	1	1	5	4
2	4	3	7	3

BEST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max
25voidmain()
{
    intfrag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;sta
    ticintbf[max],ff[max];
    clrscr();
    printf("\nEnterthenumberofblocks:");scanf("%d",&nb)
    ;
    printf("Enterthenumberoffiles:");scanf("
    %d",&nf);
```

```
    printf("\nEnter the size of the blocks:-
    \n");for(i=1;i<=nb;i++)


printf("Block%d:",i);
scanf("%d",&b[i]);
   printf("Enter the size of the files :-
   \n");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("\nFile No\tFile Size \tBlock
No\tBlockSize\tFragment");for(i=1;i
<=nf&&ff[i]!=0;i++)


printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);ge
tch();
}
```

INPUT

Enterthenumberofblocks:3
Enterthenumberoffiles: 2

Enterthesizeoftheblocks:-
Block1:5
Block2: 2
Block3: 7

Enterthesizeofthefiles:-File
1:1
File2:4

OUTPUT

FileNo	FileSize	Block No	BlockSize	Fragment
1	1	2	2	1
2	4	1	5	1

FIRST-FIT

```
#include<stdio.h>
#include<conio.h>
#define max
25voidmain()
{
intfrag[max],b[max],f[max],i,j,nb,nf,temp,highe
st=0;staticintbf[max],ff[max];
clrscr();
printf("\n\tMemoryManagementScheme-
WorstFit");printf("\nEnterthenumberofblocks:");
```

```
 scanf("%d",&nb);
printf("Enterthenumberoffiles:");scanf("
%d",&nf);
printf("\nEnter the size of the blocks:-
 \n");for(i=1;i<=nb;i++)

{
        printf("Block%d:",i);
        scanf("%d",&b[i]);
}
printf("Enter the size of the files :-
\n");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)//ifbf[j]isnotallocated
                {
                        temp=b[j]-
                        f[i];if(temp>=
                        0)
                                if(highest<temp)
                                {
                }
        }
```

```
                    frag[i]=highest;bf[ff[i]]=1;highest=0;
            }
            ff[i]=j;highest=temp;
        }
            printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement");
            for(i=1;i<=nf;i++)
                printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);ge
            tch();
}
```

INPUTEnterthenumberofblock
s:3Enterthe numberoffiles: 2

Enterthesizeoftheblocks:-
Block1:5
Block2: 2
Block3: 7

Enterthesizeofthefiles:-File
1:1
File2:4

OUTPUT

FileNo	FileSize	Block No	BlockSize	Fragment
1	1	3	7	6
2	4	1	5	1

7. Develop a C program to simulate page replacement algorithms:
a) FIFO b) LRU

A) FIRSTINFIRSTOUTS
OURCECODE :

```
#include<stdio.h>#include<
conio.h>
int fr[3];
voidmain()
{
voiddisplay();
inti,j,page[12]={2,3,2,1,5,2,4,5,3,2,5,2};
intflag1=0,flag2=0,pf=0,frsize=3,top
=0;clrscr();
for(i=0;i<3;i++)
{
fr[i]=-1;
}
for(j=0;j<12;j++)
{
flag1=0;flag2=0;for(i=0;i<12;i++)
{
if(fr[i]==page[j])
{
flag1=1;
flag2=1;
break;
}
}
```

```
if(flag1==0)
{
for(i=0;i<frsize;i++)
{
if(fr[i]==-1)
{
fr[i]=page[j];flag2=1;break;
}
}
}
if(flag2==0)
{
fr[top]=page[j];
top++;
pf++;if(top>=f
rsize)top=0;
}
display();
}
printf("Numberofpagefaults:%d",pf+frsize);get
 ch();
}
voiddisplay()
{
int     i;
        printf("\n");for(i
=0;i<3;i++)printf("%d
\t",fr[i]);
}
```

OUTPUT:

2-1-1

23-1

23-1

231

531

521

524

524

324

324

354

352

Numberofpagefaults:9

LRU

SOURCECODE:

```
#include<stdio.h>
#include<conio.h>
intfr[3];
voidmain()
{
voiddisplay();
intp[12]={2,3,2,1,5,2,4,5,3,2,5,2},i,j,fs[3];
int
 index,k,l,flag1=0,flag2=0,pf=0,frsize=3;clr
 scr();
for(i=0;i<3;i++)
```

```
{
fr[i]=-1;
}
for(j=0;j<12;j++)
{
flag1=0,flag2=0;
for(i=0;i<3;i++)
{
if(fr[i]==p[j])
{
flag1=1;flag2=
1;break;
}
}
if(flag1==0)
{
for(i=0;i<3;i++)
{
if(fr[i]==-1)
{
fr[i]=p[j];
flag2=1;
break;
}
}
}
if(flag2==0)
{
```

```
for(i=0;i<3;i++)
fs[i]=0;
for(k=j-1,l=1;l<=frsize-1;l++,k--)
{
for(i=0;i<3;i++)
{
if(fr[i]==p[k])fs[i]=1;
}}
for(i=0;i<3;i++)
{
if(fs[i]==0)
index=i;
}
fr[index]=p[j];
pf++;
}
display();
}
printf("\nnoofpagefaults:%d",pf+frsize);get
ch();
}
voiddisplay()
{
inti;printf("\n");for(i=0;i<3;i++)
printf("\t%d",fr[i]);
}
```

OUTPUT:

```
2-1-1
23-1
```

23-1

231

251

251

254

254

354

352

352

352

Noofpage faults:7

8. Simulate following File Organization Techniques
a) Single level directory b)Two level directory

SOURCECODE :

```
#include<stdio.h>
struct
{
char
        dname[10],fname[10][10];
intfcnt;
}dir;


voidmain()
{
int   i,ch;
      charf[30];
clrscr();dir.fcnt
= 0;
printf("\nEnter  name  of  directory  --
                  ");scanf("%s",dir.dname);
while(1)
{
printf("\n\n1.CreateFile\t2.DeleteFile\t3.SearchFile \n
4. Display Files\t5. Exit\nEnter your choice --
");scanf("%d",&ch);
switch(ch)
{
case1:printf("\nEnterthenameofthefile--
");scanf("%s",dir.fname[dir.fcnt]);
dir.fcnt++;break;
case 2: printf("\nEnter the name of the file --
");scanf("%s",f);
```

```
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f,dir.fname[i])==0)
{
printf("File%sisdeleted",f);
strcpy(dir.fname[i],dir.fname[dir.fcnt-1]);
break;
}
```

```
}
if(i==dir.fcnt)
printf("File%snotfound",f);
                                    else


                                        dir.fcnt--
                                        ;break;
        case3:                          printf("\nEnterthenameofthefile--");
                                        scanf("%s",f);for(i=0;
                                        i<dir.fcnt;i++)
                                        {
                                        if(strcmp(f,dir.fname[i])==0)
                                        {
                                        printf("File%sisfound",f);bre
                                        ak;
                                        }
                                        }
                                        if(i==dir.fcnt)
                                        printf("File%snotfound",f);b
                                        reak;
        case4:                          if(dir.fcnt==0)
                                        printf("\nDirectoryEmpty");
                                        else
                                        {
                                        printf("\nThe Files are --
                                        ");for(i=0;i<dir.fcnt;i++)pri
                                        ntf("\t%s",dir.fname[i]);
                                        }
                                        break;
```

```
}
getch();
}
default:exit(0);
}
```

OUTPUT:

Enternameofdirectory--CSE
1. CreateFile2. DeleteFile3. SearchFile
4. DisplayFiles5.ExitEnteryourchoice–1

Enterthe name ofthe file--A
1. CreateFile2. DeleteFile3. SearchFile
4. DisplayFiles5.ExitEnteryourchoice–1

Enterthe name ofthe file--B
1. CreateFile2. DeleteFile3. SearchFile
4. DisplayFiles5.ExitEnteryour choice–1

Enterthe name ofthe file--C
1. CreateFile2. DeleteFile3. SearchFile
4. DisplayFiles5.ExitEnteryourchoice–4

TheFilesare--ABC
1. CreateFile2. DeleteFile3. SearchFile
4. DisplayFiles5.ExitEnteryour choice–3

Enterthenameofthefile–
ABCFileABCnotfound
1. CreateFile2. DeleteFile3. SearchFile
4. DisplayFiles5. ExitEnteryourchoice–2

Enterthenameofthefile–
BFileBisdeleted

1. CreateFile2.DeleteFile3.SearchFile

4. DisplayFiles5.ExitEnteryourchoice–5

TWOLEVELDIRECTORY

SOURCECODE :

```
#include<stdio.h>
struct
{
        char
                dname[10],fname[10][10];
        intfcnt;
}dir[10];

voidmain()
{
        int    i,ch,dcnt,k;
               charf[30],    d[30];
        clrscr();dcnt=0;
        while(1)
        {
                printf("\n\n1.CreateDirectory\t2.CreateFile\t3.DeleteFile");printf("\n4
                .SearchFile\t\t5.Display\t6.Exit\tEnteryour                choice--
                ");scanf("%d",&ch);
                switch(ch)
                {
                        case 1: printf("\nEnter name of directory --
                                ");scanf("%s",
                                                dir[dcnt].dname);
                                dir[dcnt].fcnt=0;
                                dcnt++;
                                printf("Directorycreated");break;
                        case2:printf("\nEnternameofthedirectory—
```

```
");
scanf("%s",d);
        for(i=0;i<dcnt;i++)
                if(strcmp(d,dir[i].dname)==0)
                {
   printf("Entername of the file--
   ");scanf("%s",dir[i].fname[dir[i].fcnt]);
            dir[i].fcnt++;printf("F
             ilecreated");
                }
        if(i==dcnt)
            printf("Directory%snotfound",d);b
            reak;
case3:printf("\nEnternameofthedirectory--
        ");scanf("%s",d);
        for(i=0;i<dcnt;i++)
        for(i=0;i<dcnt;i++)
        {
        if(strcmp(d,dir[i].dname)==0)
        {
                printf("Enternameofthefile--
                ");scanf("%s",f);for(k=0;k<dir[i].fcnt;
                k++)
                {
                if(strcmp(f,dir[i].fname[k])==0)
                {
                printf("File%sisdeleted",f);di
                r[i].fcnt--;
                strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);got
                ojmp;
                }
        }

        printf("File%snotfound",f);gotojmp;
}
}
```

```
        printf("Directory%snotfound",d);j
        mp:break;
case4:printf("\nEnternameofthedirectory--
        ");scanf("%s",d);
        for(i=0;i<dcnt;i++)
        {
                if(strcmp(d,dir[i].dname)==0)
                {
                        printf("Enterthenameofthefile--
                        ");scanf("%s",f);for(k=0;k<dir[i].fcnt;
                        k++)
                        {
                                if(strcmp(f,dir[i].fname[k])==0)
                                {
                                printf("File%s is found",f);gotojmp1;
                                }
                        }
```

```
                                printf("File%snotfound",f);gotojmp1;
}
}printf("Directory%snotfound",d);jmp1:break;case 5:if(dcnt==0)
                        printf("\nNoDirectory's");
                                else
                                {
                                        printf("\nDirectory\tFiles");
                                        for(i=0;i<dcnt;i++)
                                        {
                                            printf("\n%s\t\t",dir[i].dname);for(k=0;k<dir[i].fcnt;k++
                                            )printf("\t%s",dir[i].fname[k]);


                                        }
                                }
                                break;

                        default:exit(0);
                }
}
getch();
 }
```

OUTPUT

1.CreateDirectory2.CreateFile3.DeleteFile

4.SearchFile5.Display6.E
xitEnteryourchoice--1

Enternameofdirectory--DIR1Directorycreated

1.CreateDirectory2.CreateFile3.DeleteFile

4. Search File 5. Display 6. Exit Enter your choice -- 1Enternameof directory-- DIR2Directorycreated

1.CreateDirectory2.CreateFile3.DeleteFile

4. Search File 5. Display 6. Exit Enter your choice -- 2Enternameofthedirectory–DIR1

Enter name of the file
-- A1Filecreated

1.CreateDirectory2.CreateFile3.DeleteFile

4.SearchFile5.Display6.E
xitEnteryourchoice--2

Enternameofthedirectory–DIR1

Enter name of the file-
-A2Filecreated

1.Create Directory

2.Create File

3.Delete File

4.Search File

5.Display

6.Exit

Enter your choice–6

9. Develop a C program to simulate the Linked file allocation strategies.

SOURCECODE :

```
#include<st
dio.h>main
()
{
intf[50],p,i,j,k,a,st,le
n,n,c;clrscr();
for(i=0;i<50;i++)f[i]=0;
printf("Enterhowmanyblocksthatarealread
yallocated");scanf("%d",&p);
printf("\nEntertheblocksno.sthatarealreadyallocated
");for(i=0;i<p;i++)
{
scanf("%d
",&a);f[a]
=1;
}
X:
printf("Enter the starting index
block &length");
scanf("%d%d",&st,&len);
k=len;for(j=st;j<(k+st);j++)
{
if(f[j]==0)
{f[j]=1;
printf("\n%d->%d",j,f[j]);
}
else
{
printf("\n %d->file is already allocated",j);
k++;
```

```
}
}
printf("\nIfuwanttoentero
nemorefile?(yes-1/no-
0)");
scanf("%d
",&c);if(c
==1)
goto x
else

getch( );
}
```

OUTPUT:

```
Enterhowmanyblocksthatarealreadyallocated3Entertheblocksno.sthatare
alreadyallocated47Enterthestartingindexblock& length379
3->1
4-
>1fileisalreadyallocat
ed5->1
6->1
7-
>1fileisalreadyallocat
ed8->1
9->1file is already
allocated10->1
11->1
12->1
```

10. Develop a C program to simulate SCAN disk scheduling algorithm.

```
#include<st
dio.h>main
()
{
    intt[20],d[20], h, i, j, n,temp, k,
    atr[20],tot,p,sum=0;clrscr();
    printf("enterthenooftrackstobetraveresed");scanf
    ("%d"',&n);
    printf("enterthepositionofhead");sca
    nf("%d",&h);
    t[0]=0;t[1]=h;
    printf("enter the
    tracks");for(i=2;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])
  {
temp
=t[j];t
[j]=t[j
+1];t[
j+1]=t
emp;
}}}
for(i=0;i<n+2;i
++)if(t[i]==h)
        j=i;k=i;
   p=0;
   while(t[j]!=0)
   {
```

```
                    atr[p]=t
                    [j];j--
                    ;p++;
                }
                atr[p]=t[j];for(p=k+1;p<n+
                2;p++,k++)
                    atr[p]=t
                [k+1];for(j=0;j
                <n+1;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("\nAverageheadermovements:%f",(float)su
                m/n);getch();
                }
```

INPUT

Enterno.oftracks:9

Entertrackposition:55 58 60 70 18 90 150 160184

OUTPUT

Trackstraversed Difference betweentracks

150	50
160	10
184	24
90	94
70	20
60	10
58	2
55	3
18	37

Average headermovements:27.77