GR22 Regulations

II B.Tech II Semester

Operating Systems Concepts Lab

(GR22A2102)

B.Tech-Computer Science and Business System

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

Gokaraju Rangaraju Institute of Engineering and Technology

Operating Systems Concepts Lab

Course Code: GR22A2102 **L/T/P/C: 0/0/2/1** **II Year II Semester**

Course Objectives:

The Objectives of this course is to provide the student to

1. Learn basic commands and Shell programming in UNIX
2. Learn different types of CPU scheduling algorithms and demonstrate the usage of semaphores for solving synchronization problems.
3. Understand deadlock avoidance and management.
4. Understand memory management techniques and various page replacement policies.
5. Learn indexing and hashing

Course Outcomes:

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

1. Demonstrate the knowledge of UNIX using commands and shell programming
2. Evaluate the performance of different types of CPU scheduling algorithms and implement problem using semaphores.
3. Simulate Banker's algorithm for deadlock avoidance
4. Implement page replacement policies and memory allocation techniques in memory management.
5. Implement indexing and hashing strategies.

Gokaraju Rangaraju Institute of Engineering and Technology

Operating Systems Concepts Lab

Course Code: GR22A2102 L/T/P/C: 0/0/2/1 II Year II Semester

Syllabus

Task 1 Experiment Unix commands (files directory, data manipulation, network communication etc)

Task 2 Write programs using shell programming and use of vi editor

Task 3 Simulate the following Scheduling algorithms using C program

a) FCFS b) SJF c) Priority d) Round Robin

Task 4 To write a C program to implement concept of Shared memory

Task 5 Simulate Thread and Multi Thread using a C program

Task 6 To write a C program to implement concept of Inter Process Communication

Task 7 Implement an Algorithm for Dead Lock Detection in C.

Task 8 Simulate Bankers Algorithm for Deadlock Avoidance in C.

Task 9 Simulate the Readers – Writers problem using semaphores.

Task 10 To write C program to implement concepts of Memory Management:

a) Simulate First Fit b) Best Fit algorithm

Task 11 To write C program to Simulate page replacement Algorithms for memory management:

a) FIFO b) LRU

Task 12 To write a C program to implement the co9n1cept of Indexing and Hashing

Text Books:

1. Operating System Concepts Essentials. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne.

Reference Books:

1. Operating Systems: Internals and Design Principles. William Stallings.
2. Operating System: A Design-oriented Approach. Charles Patrick Crowley.
3. Operating Systems: A Modern Perspective. Gary J. Nutt.
4. Design of the Unix Operating Systems. Maurice J. Bach.
5. Understanding the Linux Kernel, Daniel Pierre Bovet, Marco Cesati.

INDEX

Task 1

Task 1: Experiment Unix commands (files directory, data manipulation, network communication etc)

$cp: used for copying files.

Syntax: $cp [options] source_file destination-file

Example: $cp f1 f2

OUTPUT:

```
$cat     f1

        This is GRIET

 $cat     f2

        This is GRIET
```

It will copy the contents of f1 to f2

Options:

a)-**f**: Force copy by removing the destination file if needed.

Syntax: $cp -f source_file destination-file

Example:$cp   -f   f1 f2

```
OUTPUT:$cat  f1

          This is CSE

    $cat     f2

          This is GRIET
```

b)-**i**: Ask the confirmation to overwrite.

Syntax: $cp -i source_file destination-file

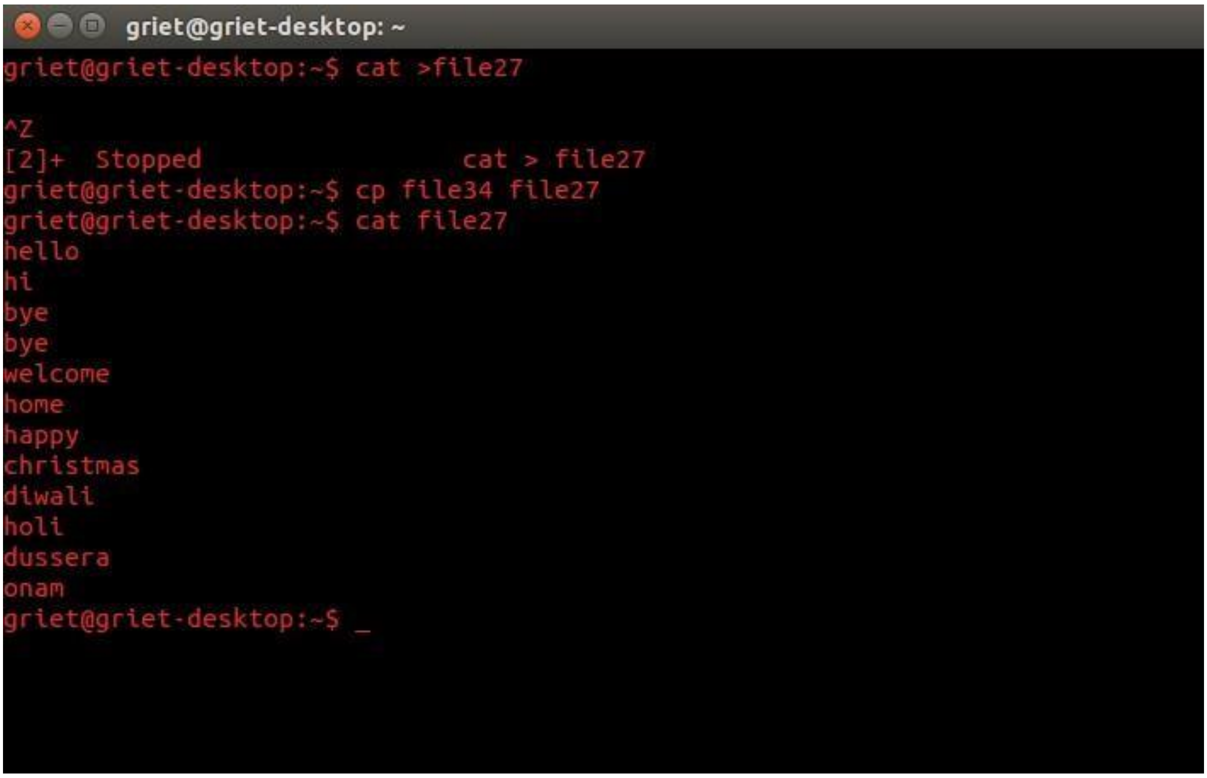
Example:$cp   -i   f1 f2

c)-**b**:It creates backup files before overriding.

Syntax: $cp -b source_file destination-file

Example:$cp -b f1 f2

Output :



griet@griet-desktop: ~

```
griet@griet-desktop:~$ cat >file27

^Z
[2]+  Stopped                 cat > file27
griet@griet-desktop:~$ cp file34 file27
griet@griet-desktop:~$ cat file27
hello
hi
bye
bye
welcome
home
happy
christmas
diwali
holi
dussera
onam
griet@griet-desktop:~$ _
```

<u>$rm:</u> Used to remove files (or) directories

Syntax: $rm [options] filename

Example:$rm f1

OUTPUT:

f1 is deleted

Options:

a)**-f**: ignores non existing files, never prompt

Syntax: $rm -f filename

Example: $rm -f myfile.txt

OUTPUT:Removes file myfile.txt

b)-**r**: Removes all files in directory and directory itself

Syntax: $rm -r filename

Example: $rm -r mydirectory

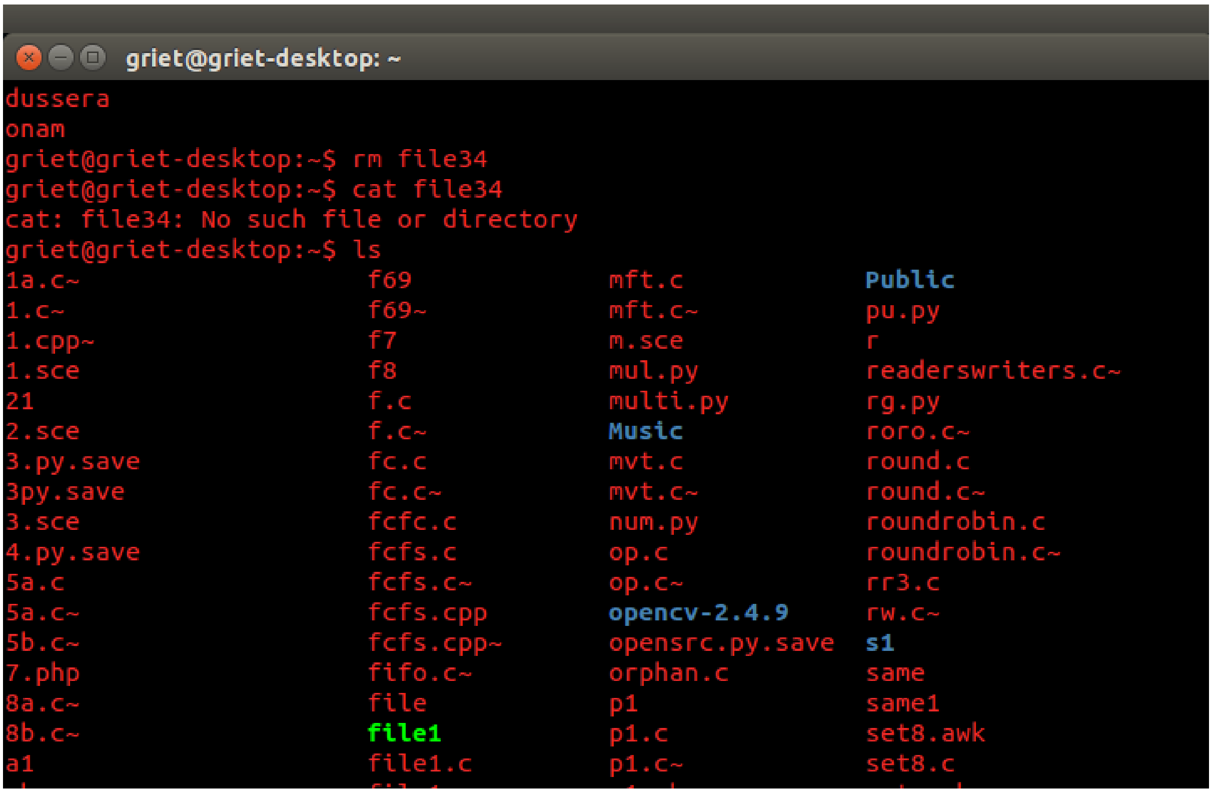
OUTPUT: Removes directory mydirectory and all files in it.

c)-**i**: prompts before every removal.

Syntax: $rm -i filename

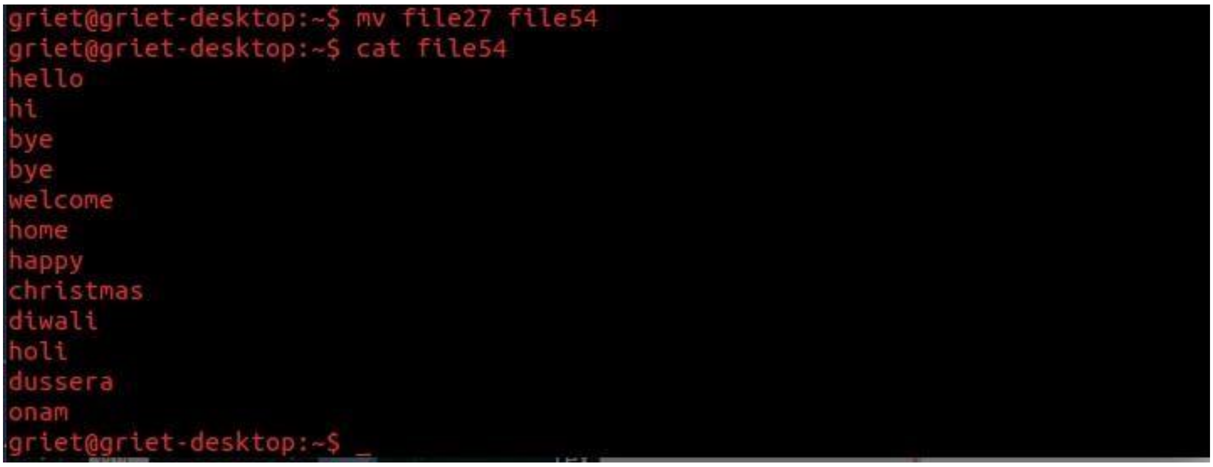
Example: $rm -i bak.c

Output:



```
dussera
onam
griet@griet-desktop:~$ rm file34
griet@griet-desktop:~$ cat file34
cat: file34: No such file or directory
griet@griet-desktop:~$ ls
1a.c~                   f69             mft.c            Public
1.c~                    f69~            mft.c~           pu.py
1.cpp~                  f7              m.sce            r
1.sce                   f8              mul.py           readerswriters.c~
21                      f.c             multi.py         rg.py
2.sce                   f.c~            Music            roro.c~
3.py.save               fc.c            mvt.c            round.c
3py.save                fc.c~           mvt.c~           round.c~
3.sce                   fcfc.c          num.py           roundrobin.c
4.py.save               fcfs.c          op.c             roundrobin.c~
5a.c                    fcfs.c~         op.c~            rr3.c
5a.c~                   fcfs.cpp        opencv-2.4.9     rw.c~
5b.c~                   fcfs.cpp~       opensrc.py.save  s1
7.php                   fifo.c~         orphan.c         same
8a.c~                   file            p1               same1
8b.c~                   file1           p1.c             set8.awk
a1                      file1.c         p1.c~            set8.c
```

$mv: mv stands for move. mv is used to move one or more files or directories from one place to another in file system like UNIX. It has two distinct functions:
(i) It rename a file or folder.
(ii) It moves group of files to different directory.
No additional space is consumed on a disk during renaming. This command normally works silently means no prompt for confirmation.
Syntax:
mv [Option] source destination

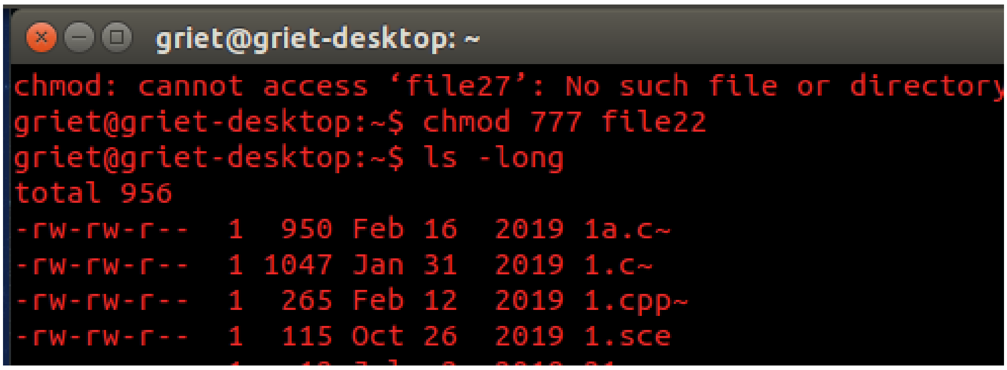


```
griet@griet-desktop:~$ mv file27 file54
griet@griet-desktop:~$ cat file54
hello
hi
bye
bye
welcome
home
happy
christmas
diwali
holi
dussera
onam
griet@griet-desktop:~$ _
```

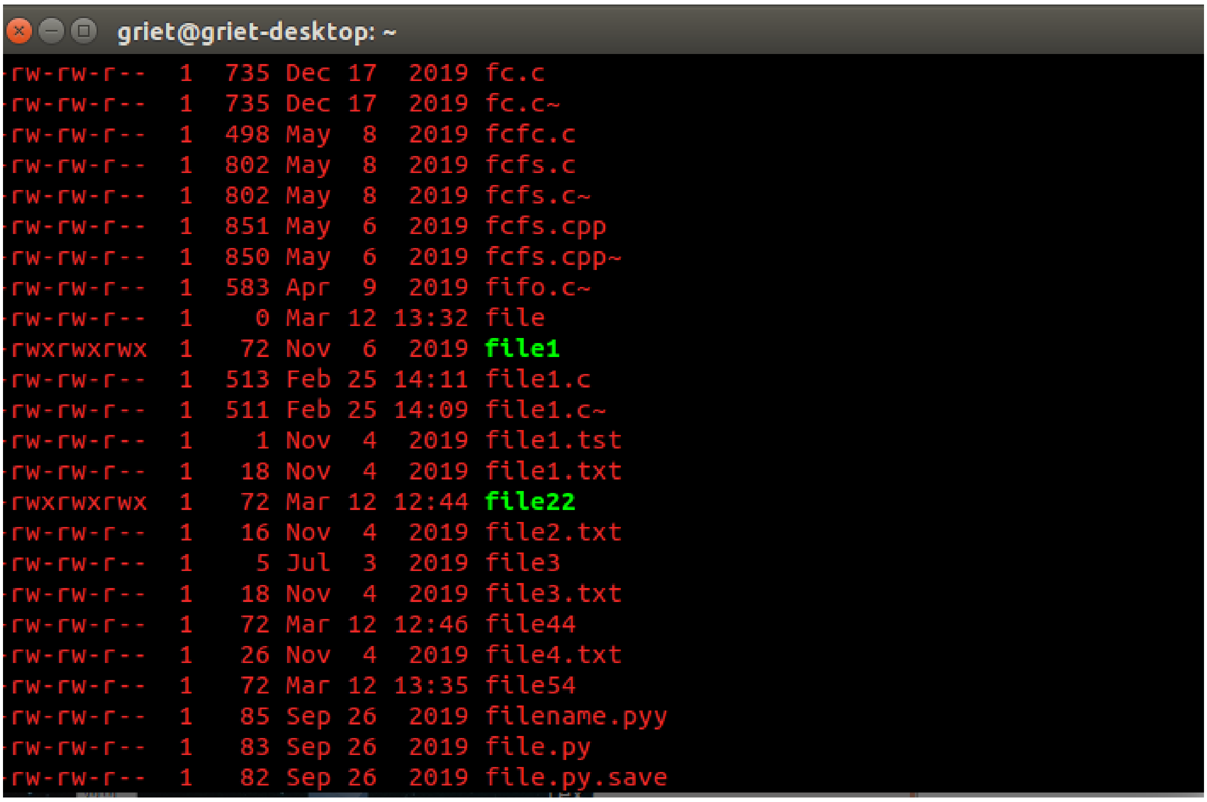
$chmod: To change directory permissions in Linux, use the following:

1. chmod +rwx filename to add permissions.
2. chmod -rwx directoryname to remove permissions.
3. chmod +x filename to allow executable permissions.
4. chmod -wx filename to take out write and executable permissions.

Output:



```
chmod: cannot access 'file27': No such file or directory
griet@griet-desktop:~$ chmod 777 file22
griet@griet-desktop:~$ ls -long
total 956
-rw-rw-r--  1  950 Feb 16  2019 1a.c~
-rw-rw-r--  1 1047 Jan 31  2019 1.c~
-rw-rw-r--  1  265 Feb 12  2019 1.cpp~
-rw-rw-r--  1  115 Oct 26  2019 1.sce
```



griet@griet-desktop: ~

```
rw-rw-r--  1  735 Dec 17  2019 fc.c
rw-rw-r--  1  735 Dec 17  2019 fc.c~
rw-rw-r--  1  498 May  8  2019 fcfc.c
rw-rw-r--  1  802 May  8  2019 fcfs.c
rw-rw-r--  1  802 May  8  2019 fcfs.c~
rw-rw-r--  1  851 May  6  2019 fcfs.cpp
rw-rw-r--  1  850 May  6  2019 fcfs.cpp~
rw-rw-r--  1  583 Apr  9  2019 fifo.c~
rw-rw-r--  1    0 Mar 12 13:32 file
rwxrwxrwx  1   72 Nov  6  2019 file1
rw-rw-r--  1  513 Feb 25 14:11 file1.c
rw-rw-r--  1  511 Feb 25 14:09 file1.c~
rw-rw-r--  1    1 Nov  4  2019 file1.tst
rw-rw-r--  1   18 Nov  4  2019 file1.txt
rwxrwxrwx  1   72 Mar 12 12:44 file22
rw-rw-r--  1   16 Nov  4  2019 file2.txt
rw-rw-r--  1    5 Jul  3  2019 file3
rw-rw-r--  1   18 Nov  4  2019 file3.txt
rw-rw-r--  1   72 Mar 12 12:46 file44
rw-rw-r--  1   26 Nov  4  2019 file4.txt
rw-rw-r--  1   72 Mar 12 13:35 file54
rw-rw-r--  1   85 Sep 26  2019 filename.pyy
rw-rw-r--  1   83 Sep 26  2019 file.py
rw-rw-r--  1   82 Sep 26  2019 file.py.save
```

$ps(Process Status):

This command is used to display the attributes of a process.

Syntax: $ps

Example: $ps

Output:
```
PID          TTY          TIME          CMD
644          01           10:30:00      bash
643          02           10:31:00      ps
```

Options:

-f: detailed listing which shows parent of every process,use(-f)->(full) option.

Example: $ps –f

Output:

```
UID     PID  PPID  C  STIME     TTY     TIME CMD
Sumid  291     1   0  10:24:36 console  0:00   –bash
```

-u:it displays processes of a user.

Example: $ps –u sumit

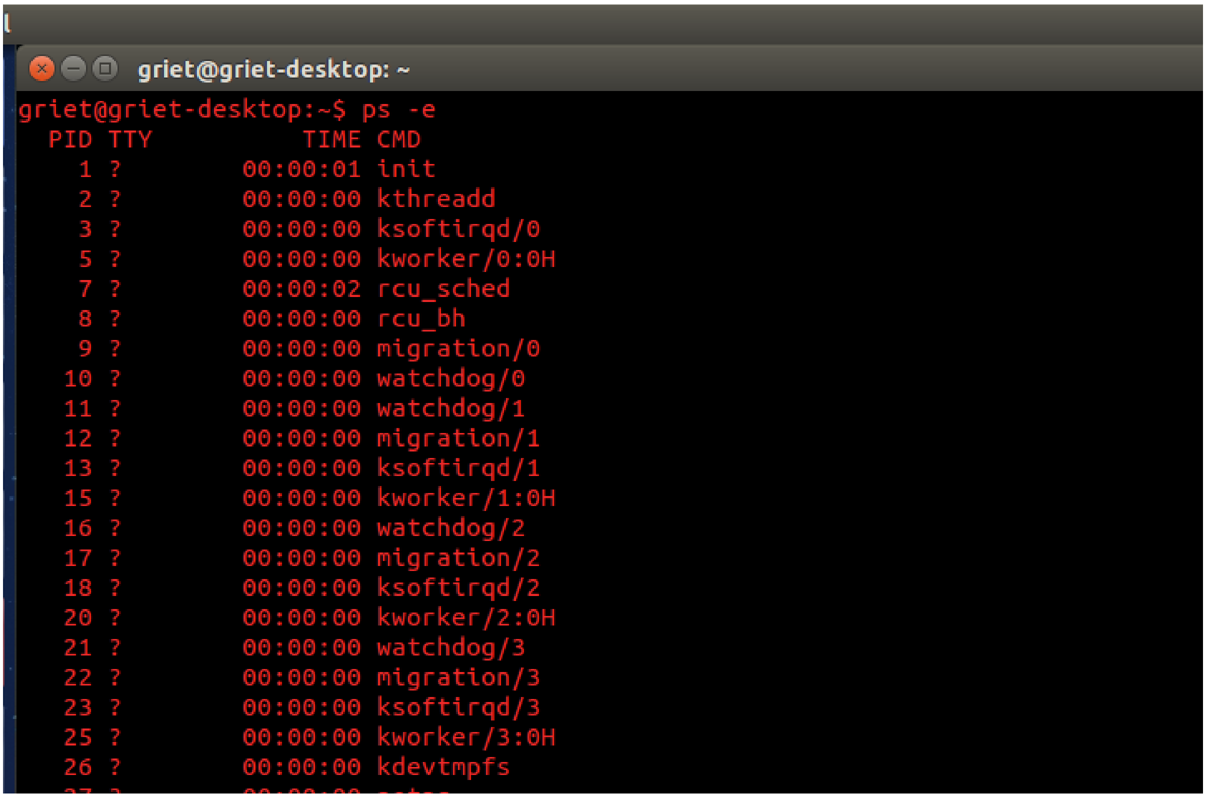
Output:
```
PID        TTY        TIME        CMD
378        ?          00:05       xsun
403        ?          00:00       xsession
```

-a: displaying all user processes.

Example: $ps –a

Output :
```
PID        TTY        TIME        CMD
662        pts/01     00:00:00    ksh
705        pts/02     00:00:00    sh
```

Output:



```
griet@griet-desktop:~$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:01 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    5 ?        00:00:00 kworker/0:0H
    7 ?        00:00:02 rcu_sched
    8 ?        00:00:00 rcu_bh
    9 ?        00:00:00 migration/0
   10 ?        00:00:00 watchdog/0
   11 ?        00:00:00 watchdog/1
   12 ?        00:00:00 migration/1
   13 ?        00:00:00 ksoftirqd/1
   15 ?        00:00:00 kworker/1:0H
   16 ?        00:00:00 watchdog/2
   17 ?        00:00:00 migration/2
   18 ?        00:00:00 ksoftirqd/2
   20 ?        00:00:00 kworker/2:0H
   21 ?        00:00:00 watchdog/3
   22 ?        00:00:00 migration/3
   23 ?        00:00:00 ksoftirqd/3
   25 ?        00:00:00 kworker/3:0H
   26 ?        00:00:00 kdevtmpfs
```

$kill: This command is used to kill the process i.e; stop or terminate a process.(by administrator)**Syntax:** $kill <pid>

Example: $kill 644

Output: The process gets terminated.

Task 2

Task: Write programs using shell programming and use of vi editor

Program:

Essential Vi Commands

- Open a file:

```
vi filename
```

- To go into edit mode:

```
press ESC and type I
```

- To go into command mode:

```
press ESC
```

- To save a file

```
press ESC and type :w fileName
```

- To save a file and quit:

```
press ESC and type :wq
```

OR

```
press ESC and type :x
```

- To jump to a line:

```
press ESC and type :the line number
```

- To Search for a string:

```
Press ESC and type /wordToSearch
```

- To quit vi:

```
Press ESC and type :q
```

Save the following into a file called hello.sh:

```
#!/bin/bash
echo "Hello, World!"
echo "Knowledge is power."
```

Save and close the file. You can run the script as follows:

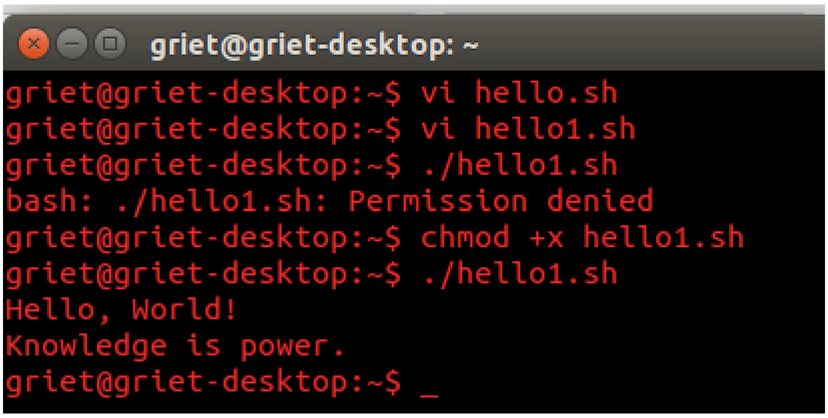
```
./hello.sh
```

Saving and Running Your Script

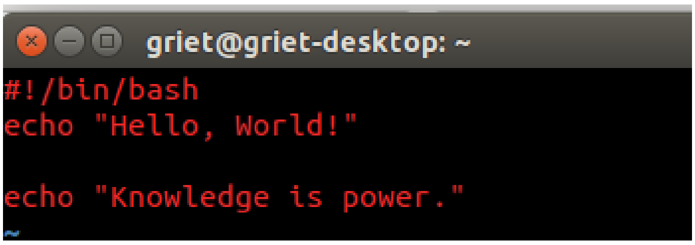
The command ./hello.sh displayed an error message on the screen. It will not run script since you've not set execute permission for your script hello.sh. To execute this program, type the following command:

```
chmod +x hello.sh
./hello.sh
```

Output:



```
griet@griet-desktop:~$ vi hello.sh
griet@griet-desktop:~$ vi hello1.sh
griet@griet-desktop:~$ ./hello1.sh
bash: ./hello1.sh: Permission denied
griet@griet-desktop:~$ chmod +x hello1.sh
griet@griet-desktop:~$ ./hello1.sh
Hello, World!
Knowledge is power.
griet@griet-desktop:~$ _
```



```
#!/bin/bash
echo "Hello, World!"

echo "Knowledge is power."
~
```

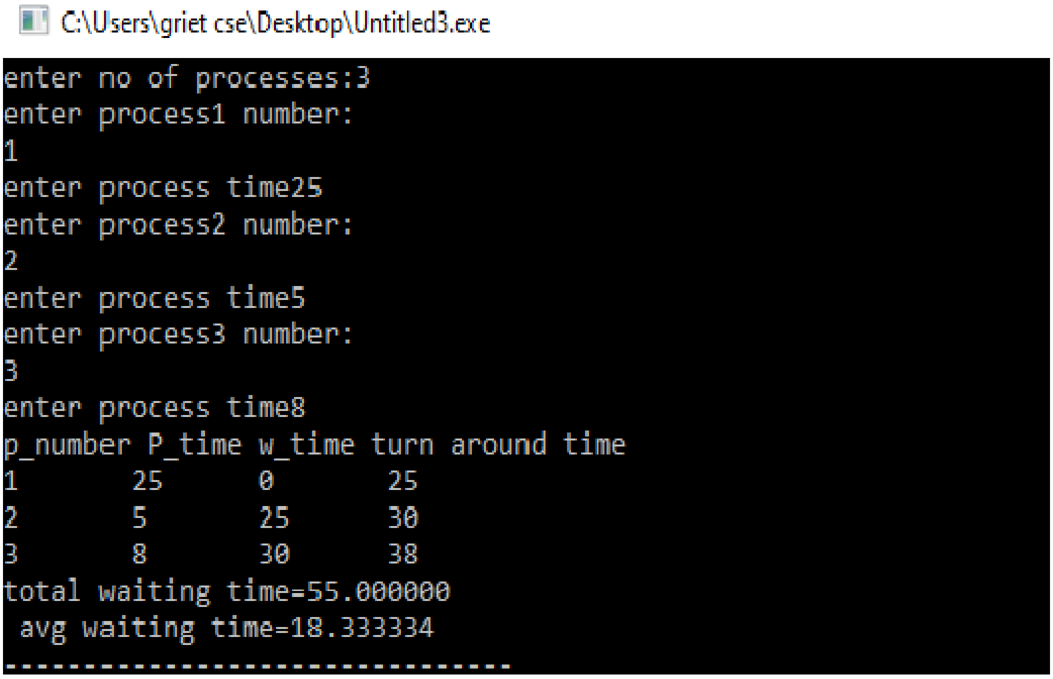
Task 3

Task 3a: Simulate the FCFS Scheduling algorithms using C program

Program:

```
#include<stdio.h>
main()
{
int p[10];
int tat[10],wt[10],i,n,pt[10],bt[10];
float avg=0,tot=0;
printf("enter no of processes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter process%d number:\n",i+1);
scanf("%d",&p[i]);
printf("enter process time");
scanf("%d",&pt[i]);
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=pt[i-1]+wt[i-1];
tot=tot+wt[i];
}
avg=(float)tot/n;
for(i=0;i<n;i++)
tat[i]=pt[i]+wt[i];
printf("p_number\t P_time\t w_time\t turn around time\n");
for(i=0;i<n;i++)
printf("%d\t%d\t%d\t%d\n",p[i],pt[i],wt[i],tat[i]);
printf("total waiting time=%f\n avg waiting time=%f",tot,avg);
}
```

Output:



C:\Users\griet cse\Desktop\Untitled3.exe

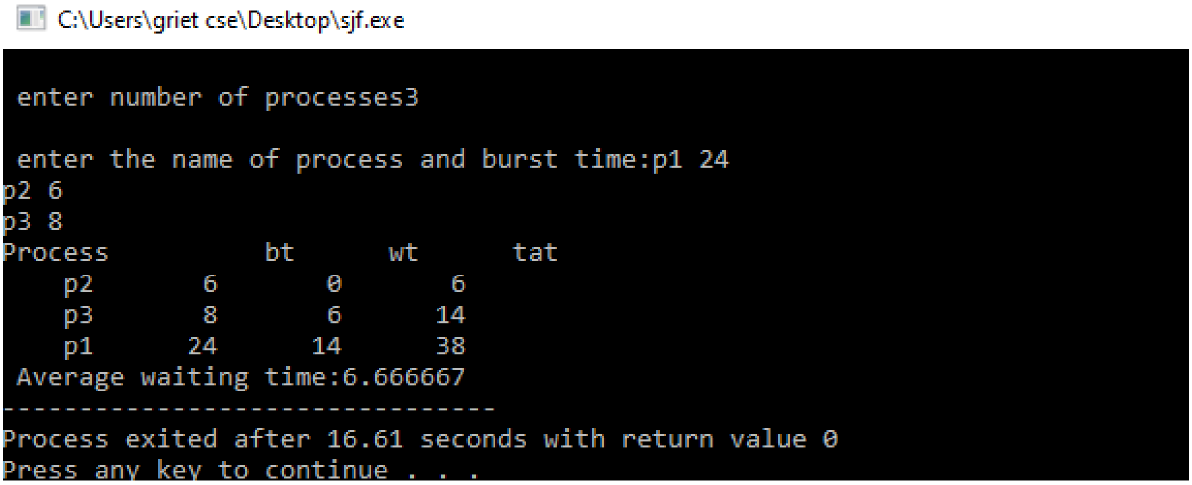
```
enter no of processes:3
enter process1 number:
1
enter process time25
enter process2 number:
2
enter process time5
enter process3 number:
3
enter process time8
p_number P_time w_time turn around time
1        25      0        25
2        5       25       30
3        8       30       38
total waiting time=55.000000
 avg waiting time=18.333334
--------------------------------
```

Task 3b: Simulate the SJF Scheduling algorithms using C program

Program:

```
#include<stdio.h>

struct sa

{

char pro[10];

int bt,wt,tat;

}p[10],temp[10];

void main()

{

int i,j,n,temp1=0;

float awt=0,atat=0;

printf("\n enter number of processes");

scanf("%d",&n);

printf("\n enter the name of process and burst time:");

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

{

scanf("%s %d",p[i].pro,&p[i].bt);

}

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

{

for(j=i+1;j<n;j++)

{

if(p[i].bt>p[j].bt)

{

temp[i]=p[i];
p[i]=p[j];
p[j]=temp[i];
}
}
}
for(i=0;i<n;i++)
{
p[i].wt=temp1;
p[i].tat=p[i].bt+p[i].wt;
temp1=p[i].bt+temp1;
}
for(i=0;i<n;i++)
{
awt=awt+p[i].wt;
}
awt=awt/n;
printf("Process \t bt \t wt \t tat");
for(i=0;i<n;i++)
{
printf("\n %5s \t %5d \t %5d \t %5d",p[i].pro,p[i].bt,p[i].wt,p[i].tat);
}
printf("\n Average waiting time:%f",awt);
}
```

Output:



C:\Users\griet cse\Desktop\sjf.exe

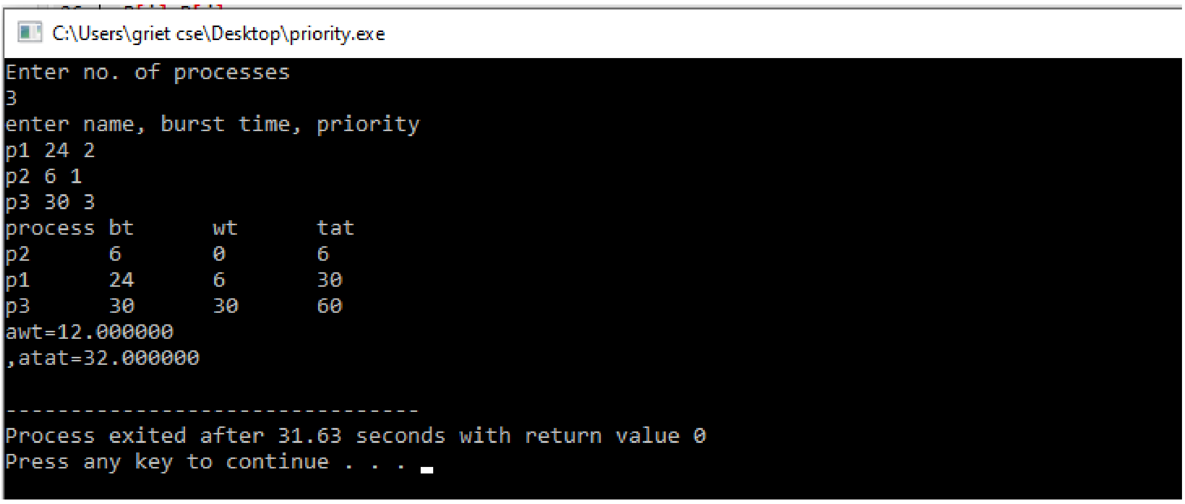
```
 enter number of processes3

 enter the name of process and burst time:p1 24
p2 6
p3 8
Process          bt       wt       tat
    p2       6        0        6
    p3       8        6       14
    p1      24       14       38
 Average waiting time:6.666667
--------------------------------
Process exited after 16.61 seconds with return value 0
Press any key to continue . . .
```

Task 3c: Simulate the Priority Scheduling algorithms using C program

Program:

```
#include<stdio.h>
struct sq
{
char pro[10];
int bt,wt,prior,tat;
}
P[10],temp;
main()
{
int i,j,n,temp1=0;
float awt=0,atat=0;
printf("Enter no. of processes\n");
scanf("%d",&n);
printf("enter name, burst time, priority\n");
for(i=0;i<n;i++)
{
scanf("%s%d%d",P[i].pro,&P[i].bt,&P[i].prior);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(P[i].prior>P[j].prior)
{
temp=P[i];
P[i]=P[j];
P[j]=temp;
}
}
}
for(i=0;i<n;i++)
{
P[i].wt=temp1;
P[i].tat=P[i].wt+P[i].bt;
temp1+=P[i].bt;
}
for(i=0;i<n;i++)
{
awt+=P[i].wt;
atat+=P[i].tat;
}
printf("process\tbt\twt\ttat\n");
awt/=n;
atat/=n;
for(i=0;i<n;i++)
{
printf("%s\t%d\t%d\t%d\n",P[i].pro,P[i].bt,P[i].wt,P[i].tat);
}
printf("awt=%f\n,atat=%f\n",awt,atat);
}
```

Output:



```
Enter no. of processes
3
enter name, burst time, priority
p1 24 2
p2 6 1
p3 30 3
process bt      wt      tat
p2      6       0       6
p1      24      6       30
p3      30      30      60
awt=12.000000
,atat=32.000000

--------------------------------
Process exited after 31.63 seconds with return value 0
Press any key to continue . . .
```

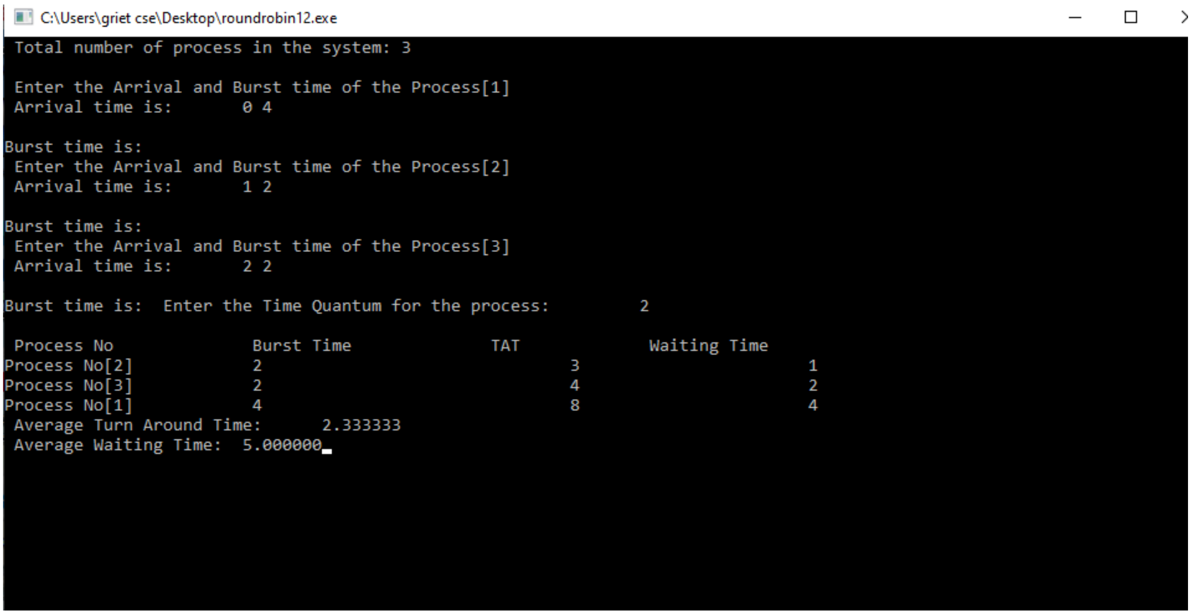
Task 3d: Simulate the Round Robin Scheduling algorithms using C program

Program:

```
#include<stdio.h>
#include<conio.h>

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

// Use for loop to enter the details of the process like Arrival time and the Burst Time
for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t"); // Accept arrival time
scanf("%d", &at[i]);
printf(" \nBurst time is: \t"); // Accept the Burst time
scanf("%d", &bt[i]);
temp[i] = bt[i]; // store the burst time in temp array
}
// Accept the Time qunat
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
// Display the process No, burst time, Turn Around Time and the waiting time
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0) // define the conditions
{
   sum = sum + temp[i];
   temp[i] = 0;
   count=1;
   }
   else if(temp[i] > 0)
   {
      temp[i] = temp[i] - quant;
      sum = sum + quant;
   }
   if(temp[i]==0 && count==1)
   {
      y--; //decrement the process no.
      printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-
bt[i]);
      wt = wt+sum-at[i]-bt[i];
      tat = tat+sum-at[i];
      count =0;
   }
   if(i==NOP-1)
   {
      i=0;
   }
   else if(at[i+1]<=sum)
   {
      i++;
   }
   else
   {
      i=0;
   }
}
```

```
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
getch();
}
```

Output:



```
 Total number of process in the system: 3

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:      0 4

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

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

Burst time is:  Enter the Time Quantum for the process:         2

 Process No              Burst Time              TAT             Waiting Time
Process No[2]            2                               3                       1
Process No[3]            2                               4                       2
Process No[1]            4                               8                       4
 Average Turn Around Time:      2.333333
 Average Waiting Time:  5.000000
```

Task 4

Task 4: To write a C program to implement concept of Shared memory

Program:

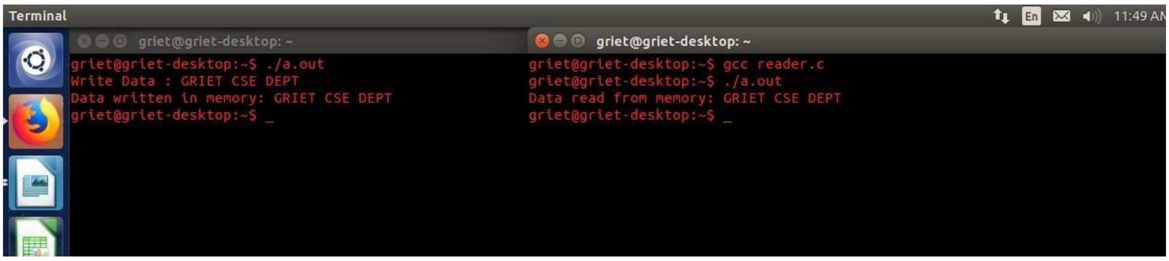
Writer

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <string.h>
int main()
{
    // ftok to generate unique key
    key_t key = ftok("shmfile",65);
      // shmget returns an identifier in shmid
    int shmid = shmget(key,1024,0666|IPC_CREAT);
      // shmat to attach to shared memory
    char *str = (char*) shmat(shmid,(void*)0,0);
      printf("Write Data : ");
    gets(str);
      printf("Data written in memory: %s\n",str);
          //detach from shared memory
    shmdt(str);
      return 0;
}
```

Reader

```
#include <sys/ipc.h>
#include <sys/shm.h>
#include  <stdio.h>
int main()
{
        // ftok to generate unique key
        key_t key = ftok("shmfile",65);

        // shmget returns an identifier in shmid
        int shmid = shmget(key,1024,0666|IPC_CREAT);

        // shmat to attach to shared memory
        char *str = (char*) shmat(shmid,(void*)0,0);

        printf("Data read from memory: %s\n",str);

        //detach from shared memory
        shmdt(str);

        // destroy the shared memory
        shmctl(shmid,IPC_RMID,NULL);

        return 0;
}
```

Output:



```
griet@griet-desktop:~$ ./a.out
Write Data : GRIET CSE DEPT
Data written in memory: GRIET CSE DEPT
griet@griet-desktop:~$ _
```

```
griet@griet-desktop:~$ gcc reader.c
griet@griet-desktop:~$ ./a.out
Data read from memory: GRIET CSE DEPT
griet@griet-desktop:~$ _
```

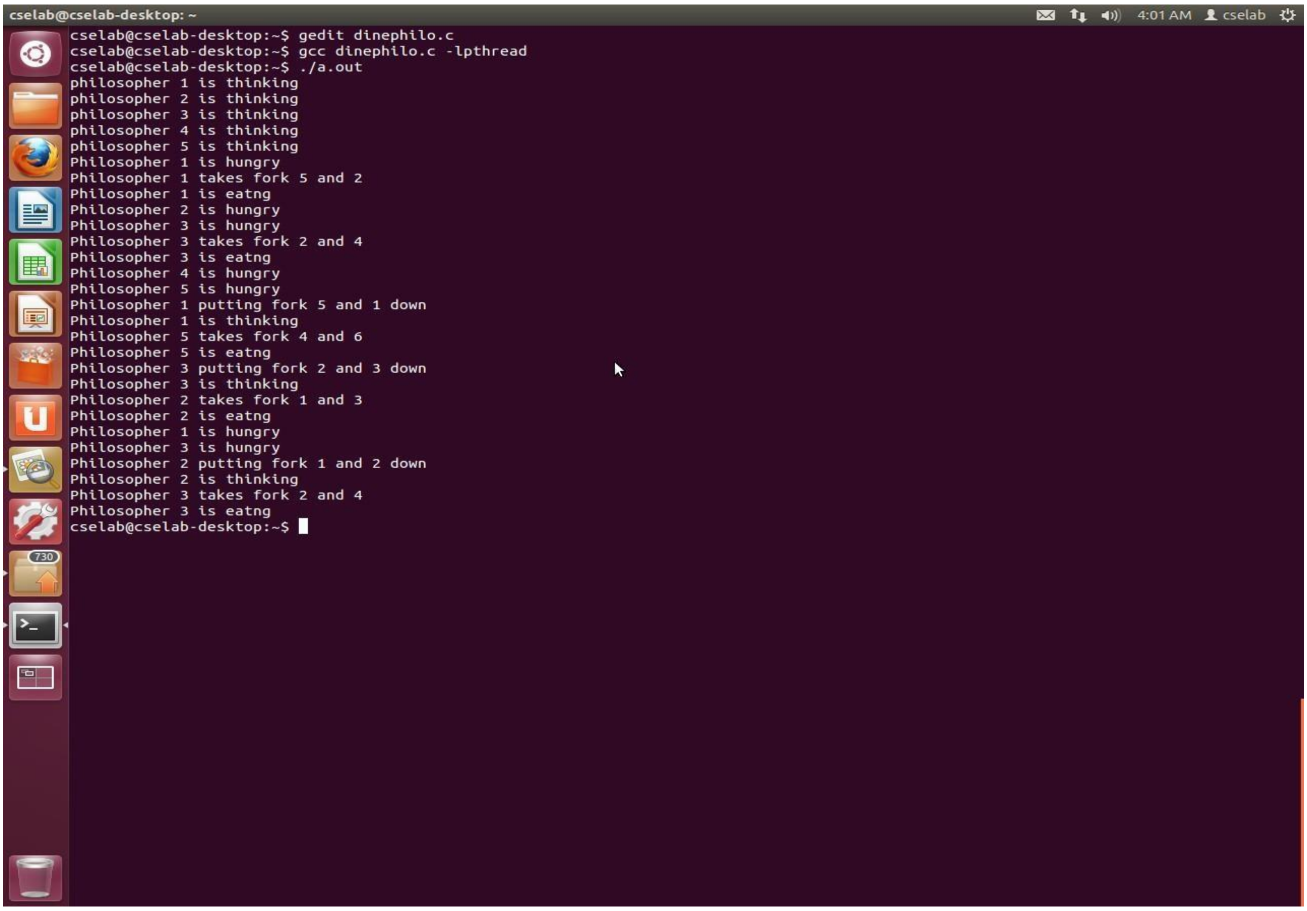
Task 5

Task 5: Simulate Thread and Multi Thread using a C program

Program:

```
// Program to implement Dining Philosopher problem using semaphores.

#include<stdio.h>
#include<stdlib.h>
#include<semaphore.h>
#define N 5
#define thinking 0
#define hungry 1
#define eating 2
#define left (ph_num+4)%N
#define right (ph_num+1)%N
sem_t mutex;
sem_t s[N];
void *philosopher(void *num);
void take_fork(int);
void put_fork(int);
void teet(int);
int state[N]={thinking,thinking,thinking,thinking,thinking};
int phil_num[N]={0,1,2,3,4};
int main()
{
int i;
pthread_t thread_id[N];
sem_init(&mutex,0,1);
for(i=0;i<N;i++)
sem_init(&s[i],0,0);
for(i=0;i<N;i++)
{
pthread_create(&thread_id[i],NULL,philosopher,&phil_num[i]);
printf("philosopher %d is thinking \n",i+1);
}
for(i=0;i<N;i++)
pthread_join(thread_id[i],NULL);
}
void *philosopher(void *num)
{
while(1)
{
int *i=num;
sleep(1);
take_fork(*i);
sleep(1);
put_fork(*i);
}
}
void take_fork(int ph_num)
{
sem_wait(&mutex);
state[ph_num]=hungry;
printf("Philosopher %d is hungry\n",ph_num+1);
teet(ph_num);
sem_post(&mutex);
sem_wait(&s[ph_num]);
sleep(1);
}
void teet(int ph_num)
{
static count=0;
if(state[ph_num]==hungry&& state[left]!=eating && state[right]!=eating)
{
state[ph_num]=eating;
printf("Philosopher %d takes fork %d and %d\n",ph_num+1,left+1,ph_num+2);
printf("Philosopher %d is eatng\n",ph_num+1);
sem_post(&s[ph_num]);
count++;
}
if(count==5)
exit(1);
}
void put_fork(int ph_num)
{
sem_wait(&mutex);
state[ph_num]=thinking;
printf("Philosopher %d putting fork %d and %d down \n",ph_num+1,left+1,ph_num+1);
printf("Philosopher %d is thinking\n",ph_num+1);
teet(left);
teet(right);
sem_post(&mutex);
}
```

Output:



```
cselab@cselab-desktop:~$ gedit dinephilo.c
cselab@cselab-desktop:~$ gcc dinephilo.c -lpthread
cselab@cselab-desktop:~$ ./a.out
philosopher 1 is thinking
philosopher 2 is thinking
philosopher 3 is thinking
philosopher 4 is thinking
philosopher 5 is thinking
Philosopher 1 is hungry
Philosopher 1 takes fork 5 and 2
Philosopher 1 is eatng
Philosopher 2 is hungry
Philosopher 3 is hungry
Philosopher 3 takes fork 2 and 4
Philosopher 3 is eatng
Philosopher 4 is hungry
Philosopher 5 is hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 6
Philosopher 5 is eatng
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 3
Philosopher 2 is eatng
Philosopher 1 is hungry
Philosopher 3 is hungry
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 3 takes fork 2 and 4
Philosopher 3 is eatng
cselab@cselab-desktop:~$ █
```

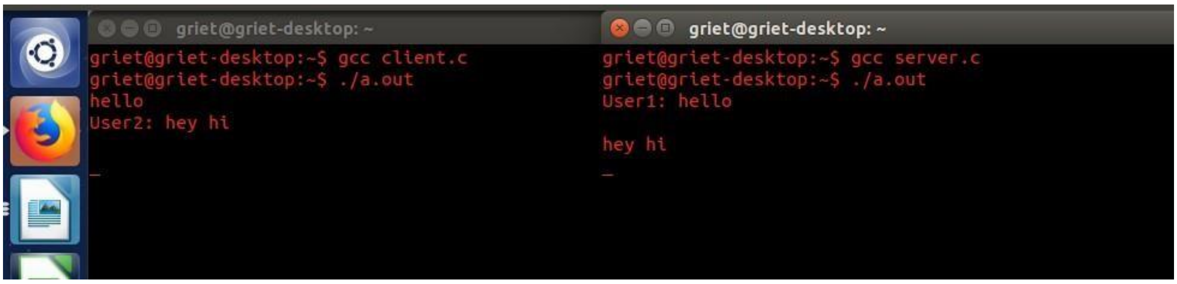
Task 6

Task: To write a C program to implement concept of Inter Process Communication

Program:

```
// C program to implement one side of FIFO
// This side writes first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include  <unistd.h>
int main()
{
        int fd;
        // FIFO file path
        char * myfifo = "/tmp/myfifo";
        // Creating the named file(FIFO)
        // mkfifo(<pathname>, <permission>)
        mkfifo(myfifo, 0666);
        char arr1[80], arr2[80];
        while (1)
        {
        // Open FIFO for write only
        fd = open(myfifo, O_WRONLY);
        // Take an input arr2ing from user.
        // 80 is maximum length
        fgets(arr2, 80, stdin);
        // Write the input arr2ing on FIFO
        // and close it
        write(fd, arr2, strlen(arr2)+1);
        close(fd);
        // Open FIFO for Read only
        fd = open(myfifo, O_RDONLY);
        // Read from FIFO
        read(fd, arr1, sizeof(arr1));
        // Print the read message
        printf("User2: %s\n", arr1);
        close(fd);
        }
        return 0;
}

// C program to implement one side of FIFO
// This side reads first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include  <unistd.h>
int main()
{
        int fd1;
        // FIFO file path
        char * myfifo = "/tmp/myfifo";
        // Creating the named file(FIFO)
        // mkfifo(<pathname>,<permission>)
        mkfifo(myfifo, 0666);
        char str1[80], str2[80];
        while (1)
        {
        // First open in read only and read
        fd1 = open(myfifo,O_RDONLY);
        read(fd1, str1, 80);
        // Print the read string and close
        printf("User1: %s\n", str1);
        close(fd1);
        // Now open in write mode and write
        // string taken from user.
        fd1 = open(myfifo,O_WRONLY);
        fgets(str2, 80, stdin);
        write(fd1, str2, strlen(str2)+1);
        close(fd1);
        }
        return 0;
}
```

Output:



```
griet@griet-desktop:~$ gcc client.c
griet@griet-desktop:~$ ./a.out
hello
User2: hey hi

_
```

```
griet@griet-desktop:~$ gcc server.c
griet@griet-desktop:~$ ./a.out
User1: hello

hey hi

_
```

Task 7

Task: Implement an Algorithm for Dead Lock Detection in C

Program:

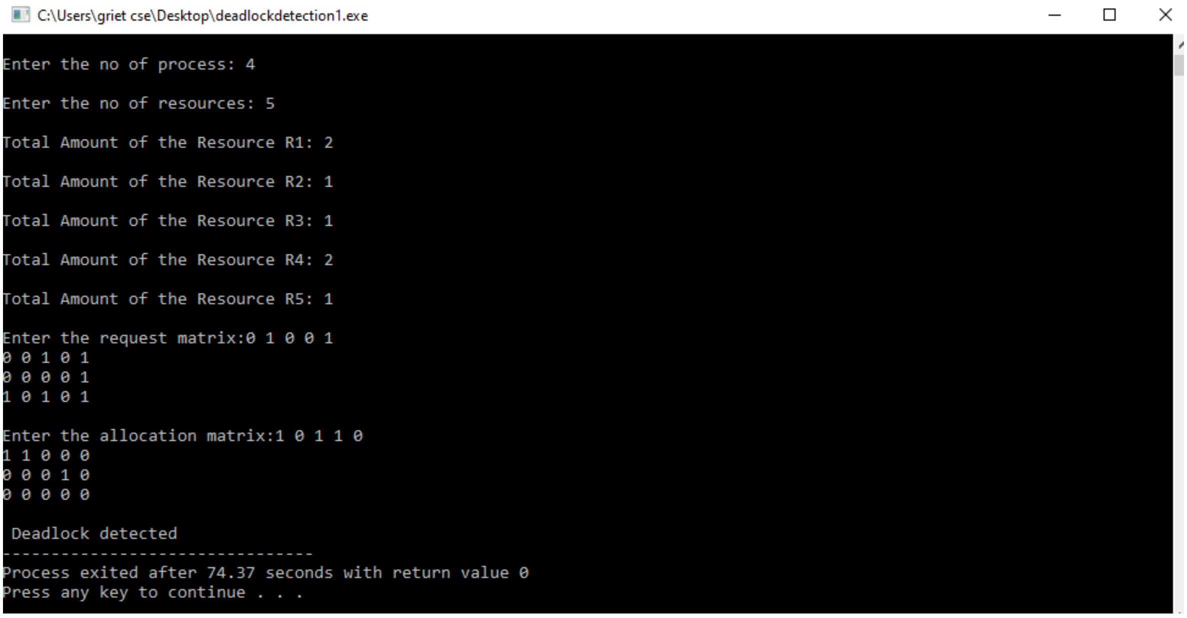
```
#include<stdio.h>

static int mark[20];

int i,j,np,nr;

int main()

{

int alloc[10][10],request[10][10],avail[10],r[10],w[10];

printf("\nEnter the no of process: ");

scanf("%d",&np);

printf("\nEnter the no of resources: ");

scanf("%d",&nr);

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

{

printf("\nTotal Amount of the Resource R%d: ",i+1);

scanf("%d",&r[i]);

}

printf("\nEnter the request matrix:");

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

for(j=0;j<nr;j++)

scanf("%d",&request[i][j]);

printf("\nEnter the allocation matrix:");

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

for(j=0;j<nr;j++)

scanf("%d",&alloc[i][j]);

/*Available Resource calculation*/
```

```
for(j=0;j<nr;j++)
{
avail[j]=r[j];
for(i=0;i<np;i++)
{
avail[j]-=alloc[i][j];
}
}
//marking processes with zero allocation
for(i=0;i<np;i++)
{
int count=0;
 for(j=0;j<nr;j++)
   {
      if(alloc[i][j]==0)
        count++;
      else
        break;
    }
 if(count==nr)
 mark[i]=1;
}
// initialize W with avail
for(j=0;j<nr;j++)
    w[j]=avail[j];
//mark processes with request less than or equal to W
for(i=0;i<np;i++)
```

```
{
int canbeprocessed=0;
 if(mark[i]!=1)
{
   for(j=0;j<nr;j++)
    {
      if(request[i][j]<=w[j])
      canbeprocessed=1;
      else
         {
         canbeprocessed=0;
         break;
          }
     }
if(canbeprocessed)
{
mark[i]=1;
for(j=0;j<nr;j++)
w[j]+=alloc[i][j];
}
}
}
//checking for unmarked processes
int deadlock=0;
for(i=0;i<np;i++)
if(mark[i]!=1)
deadlock=1;
```

```
if(deadlock)

printf("\n Deadlock detected");

else

printf("\n No Deadlock possible");

}
```

Output:



C:\Users\griet cse\Desktop\deadlockdetection1.exe

```
Enter the no of process: 4

Enter the no of resources: 5

Total Amount of the Resource R1: 2

Total Amount of the Resource R2: 1

Total Amount of the Resource R3: 1

Total Amount of the Resource R4: 2

Total Amount of the Resource R5: 1

Enter the request matrix:0 1 0 0 1
0 0 1 0 1
0 0 0 0 1
1 0 1 0 1

Enter the allocation matrix:1 0 1 1 0
1 1 0 0 0
0 0 0 1 0
0 0 0 0 0

 Deadlock detected
--------------------------------
Process exited after 74.37 seconds with return value 0
Press any key to continue . . .
```

Task 8

Task 8 Simulate Bankers Algorithm for Deadlock Avoidance in C.

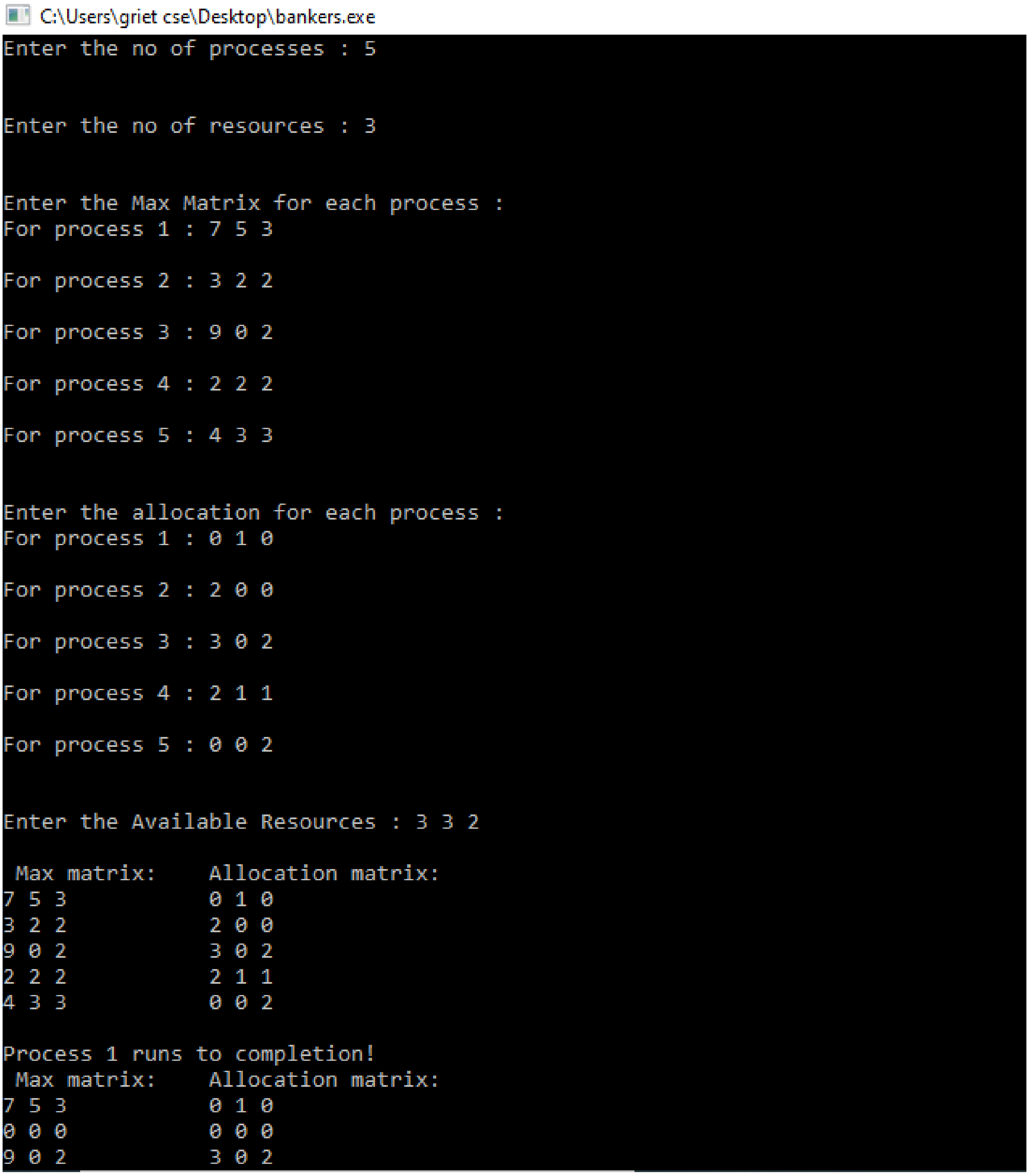
Program:

```
#include  <stdio.h>
#include <stdlib.h>
int main()
{
int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];
/*Max denotes max required resource
alloc denotes already allocated resouces for each process
avail denotes available resource of each kind
completed array indicates whether each process has met with its requirements and completed
or not.
Safe sequence is an array which holds order of execution that can result in completion of all
process*/
int p, r, i, j, process, count;
count = 0;
printf("Enter the no of processes : ");
scanf("%d", &p);
for(i = 0; i< p; i++)
completed[i] = 0; /*initially no process is completed*/
printf("\n\nEnter the no of resources : ");
scanf("%d", &r);
printf("\n\nEnter the Max Matrix for each process : ");
for(i = 0; i < p; i++)
{
printf("\nFor process %d : ", i + 1);
for(j = 0; j < r; j++)
scanf("%d", &Max[i][j]);
}
printf("\n\nEnter the allocation for each process : ");
for(i = 0; i < p; i++)
{
printf("\nFor process %d : ",i + 1);
for(j = 0; j < r; j++)
scanf("%d", &alloc[i][j]);
}
printf("\n\nEnter the Available Resources : ");
for(i = 0; i < r; i++)
scanf("%d", &avail[i]);
for(i = 0; i < p; i++)
for(j = 0; j < r; j++)
need[i][j] = Max[i][j] - alloc[i][j]; // process still need these many resorces.
do
{
printf("\n Max matrix:\tAllocation matrix:\n");
for(i = 0; i < p; i++)
{
```

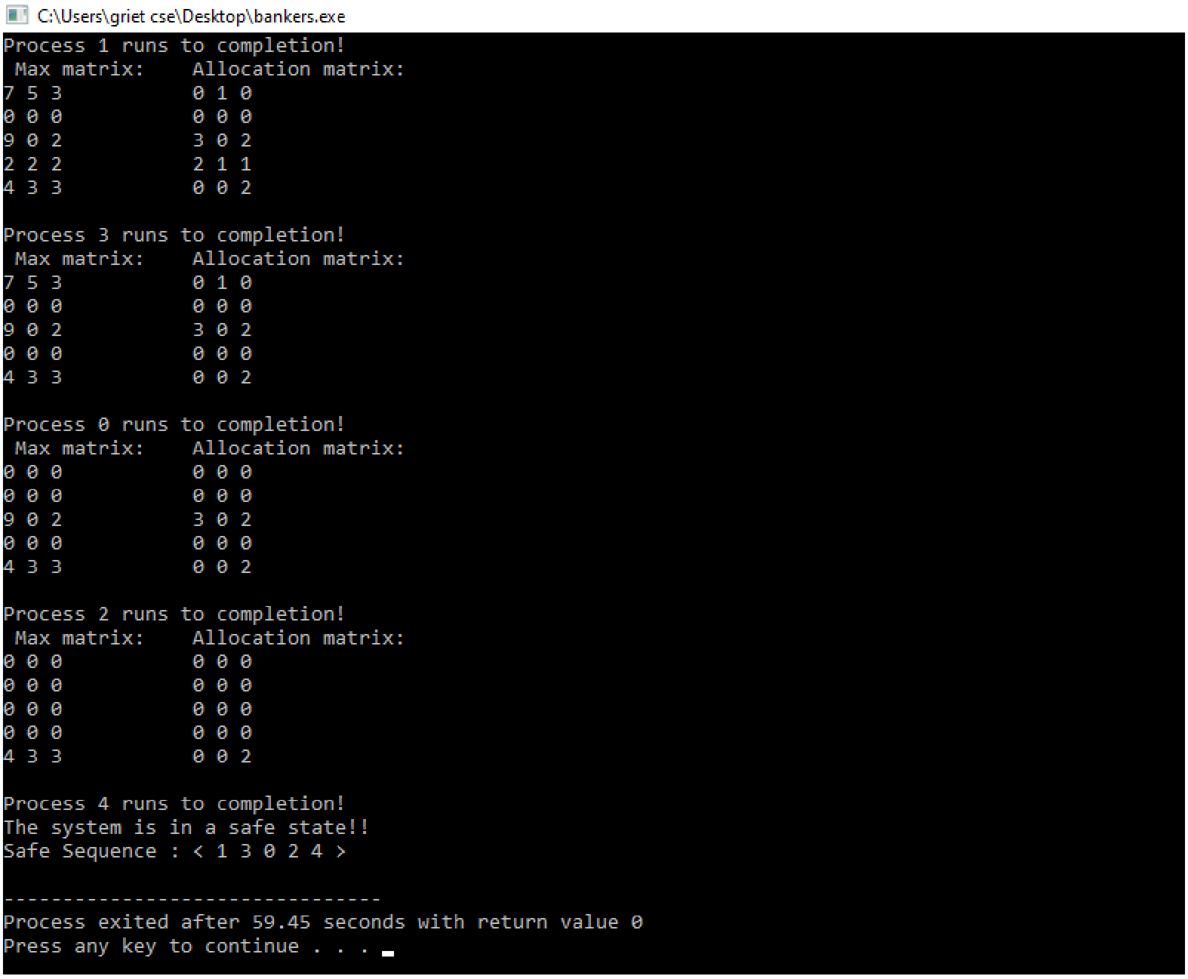
```
for( j = 0; j < r; j++)
printf("%d ", Max[i][j]);
printf("\t\t");
for( j = 0; j < r; j++)
printf("%d ", alloc[i][j]);
printf("\n");
}
process = -1; //indicates process can not completed.
        for(i = 0; i < p; i++)
        {
        if(completed[i] == 0)//if not completed.
        {
        process = i ; //ith process not yet completed.
        for(j = 0; j < r; j++)
           {
        if(avail[j] < need[i][j])
              {
        process = -1; //excess required which is not possible
        break;
              }
           }
        }/*end if*/
if(process != -1)
break; /* that means there exists a process that can complete its requirement*/
        }/*for end*/
/* process holds i th process which is not yet completed*/
if(process != -1)
{
printf("\nProcess %d runs to completion!", process );
safeSequence[count] = process ; /*join it to safe sequence*/
count++;    //identifying number of completed processes
for(j = 0; j < r; j++)
{
avail[j] += alloc[process][j]; /*return back the resources*/
alloc[process][j] = 0;
Max[process][j] = 0;
completed[process] = 1;
}
}
}while(count != p && process != -1); /*for all process*/

if(count == p)
{
printf("\nThe system is in a safe state!!\n");
printf("Safe Sequence : < ");
for( i = 0; i < p; i++)
printf("%d ", safeSequence[i]);
printf(">\n");
}
else
```

```
printf("\nThe system is in an unsafe state!!");
}
```

Output:



```
Enter the no of processes : 5


Enter the no of resources : 3


Enter the Max Matrix for each process :
For process 1 : 7 5 3

For process 2 : 3 2 2

For process 3 : 9 0 2

For process 4 : 2 2 2

For process 5 : 4 3 3


Enter the allocation for each process :
For process 1 : 0 1 0

For process 2 : 2 0 0

For process 3 : 3 0 2

For process 4 : 2 1 1

For process 5 : 0 0 2


Enter the Available Resources : 3 3 2

 Max matrix:    Allocation matrix:
7 5 3           0 1 0
3 2 2           2 0 0
9 0 2           3 0 2
2 2 2           2 1 1
4 3 3           0 0 2

Process 1 runs to completion!
 Max matrix:    Allocation matrix:
7 5 3           0 1 0
0 0 0           0 0 0
9 0 2           3 0 2
```



```
Process 1 runs to completion!
 Max matrix:    Allocation matrix:
7 5 3           0 1 0
0 0 0           0 0 0
9 0 2           3 0 2
2 2 2           2 1 1
4 3 3           0 0 2

Process 3 runs to completion!
 Max matrix:    Allocation matrix:
7 5 3           0 1 0
0 0 0           0 0 0
9 0 2           3 0 2
0 0 0           0 0 0
4 3 3           0 0 2

Process 0 runs to completion!
 Max matrix:    Allocation matrix:
0 0 0           0 0 0
0 0 0           0 0 0
9 0 2           3 0 2
0 0 0           0 0 0
4 3 3           0 0 2

Process 2 runs to completion!
 Max matrix:    Allocation matrix:
0 0 0           0 0 0
0 0 0           0 0 0
0 0 0           0 0 0
0 0 0           0 0 0
4 3 3           0 0 2

Process 4 runs to completion!
The system is in a safe state!!
Safe Sequence : < 1 3 0 2 4 >

--------------------------------
Process exited after 59.45 seconds with return value 0
Press any key to continue . . .
```

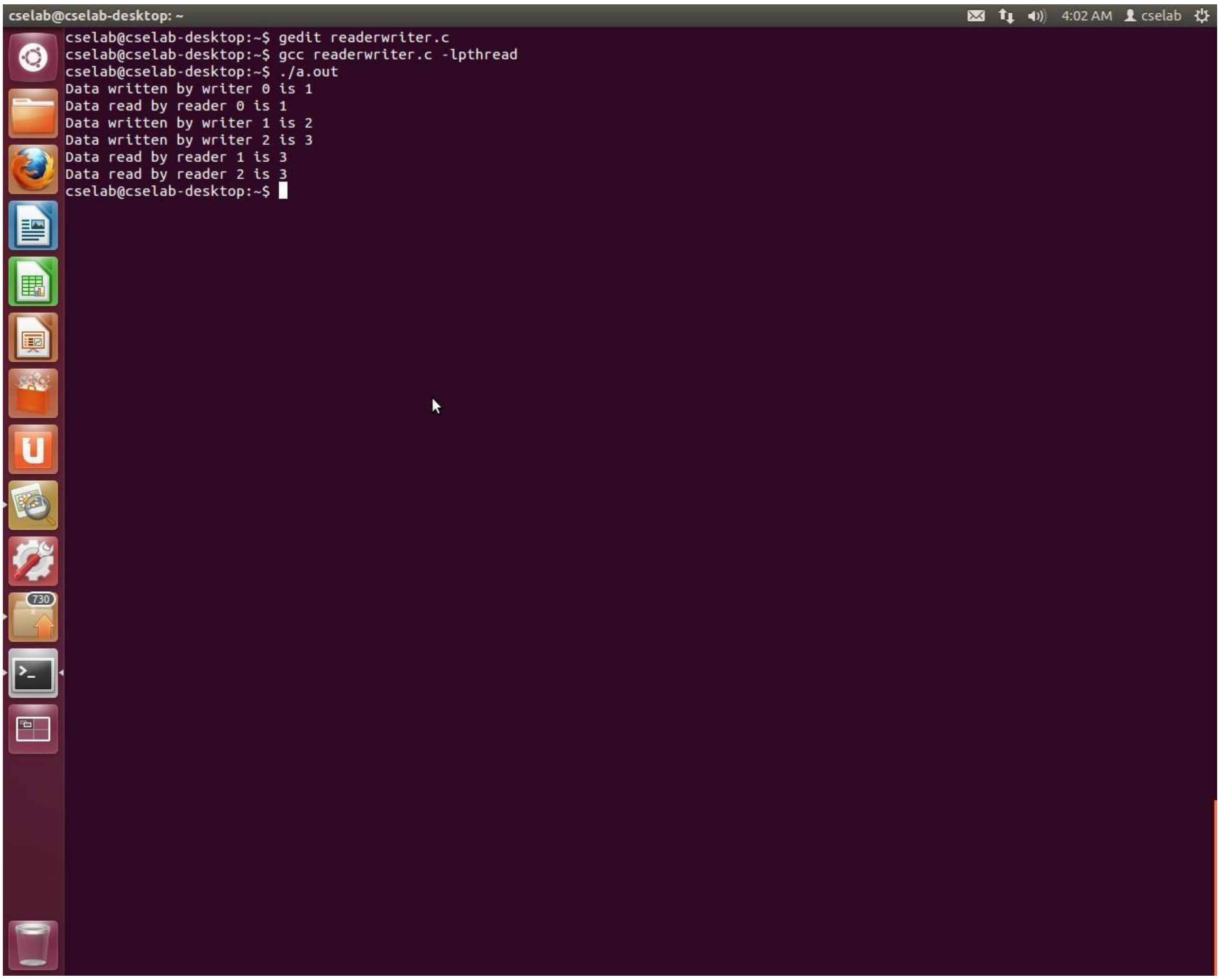
Task 9

Task: Simulate the Readers –Writers problem using semaphores

Program:

```
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>
sem_t mutex,writeblock;
int data=0,rcount=0;
void *reader(void *arg)
{
int f;
f=((int)arg);
sem_wait(&mutex);
rcount=rcount+1;
if(rcount==1)
{
sem_wait(&writeblock);
sem_post(&mutex);
printf("Data read by reader %d is %d\n",f,data);
sem_wait(&mutex);
rcount=rcount-1;
}
if(rcount==0)
{
sem_post(&writeblock);
sem_post(&mutex);
}
}
void *writer(void *arg)
{
int f;
f=((int)arg);
sem_wait(&writeblock);
data++;
printf("Data written by writer %d is %d\n",f,data);
sleep(1);
sem_post(&writeblock);
}
void main()
{
int i,b;
pthread_t rtid[5],wtid[5];
sem_init(&mutex,0,1);
sem_init(&writeblock,0,1);
for(i=0;i<=2;i++)
{
pthread_create(&wtid[i],NULL,writer,(void*)i);
pthread_create(&rtid[i],NULL,reader,(void*)i);
}
for(i=0;i<=2;i++)
{
pthread_join(wtid[i],NULL);
pthread_join(rtid[i],NULL);
}
}
```

Output:



```
cselab@cselab-desktop:~$ gedit readerwriter.c
cselab@cselab-desktop:~$ gcc readerwriter.c -lpthread
cselab@cselab-desktop:~$ ./a.out
Data written by writer 0 is 1
Data read by reader 0 is 1
Data written by writer 1 is 2
Data written by writer 2 is 3
Data read by reader 1 is 3
Data read by reader 2 is 3
cselab@cselab-desktop:~$ 
```

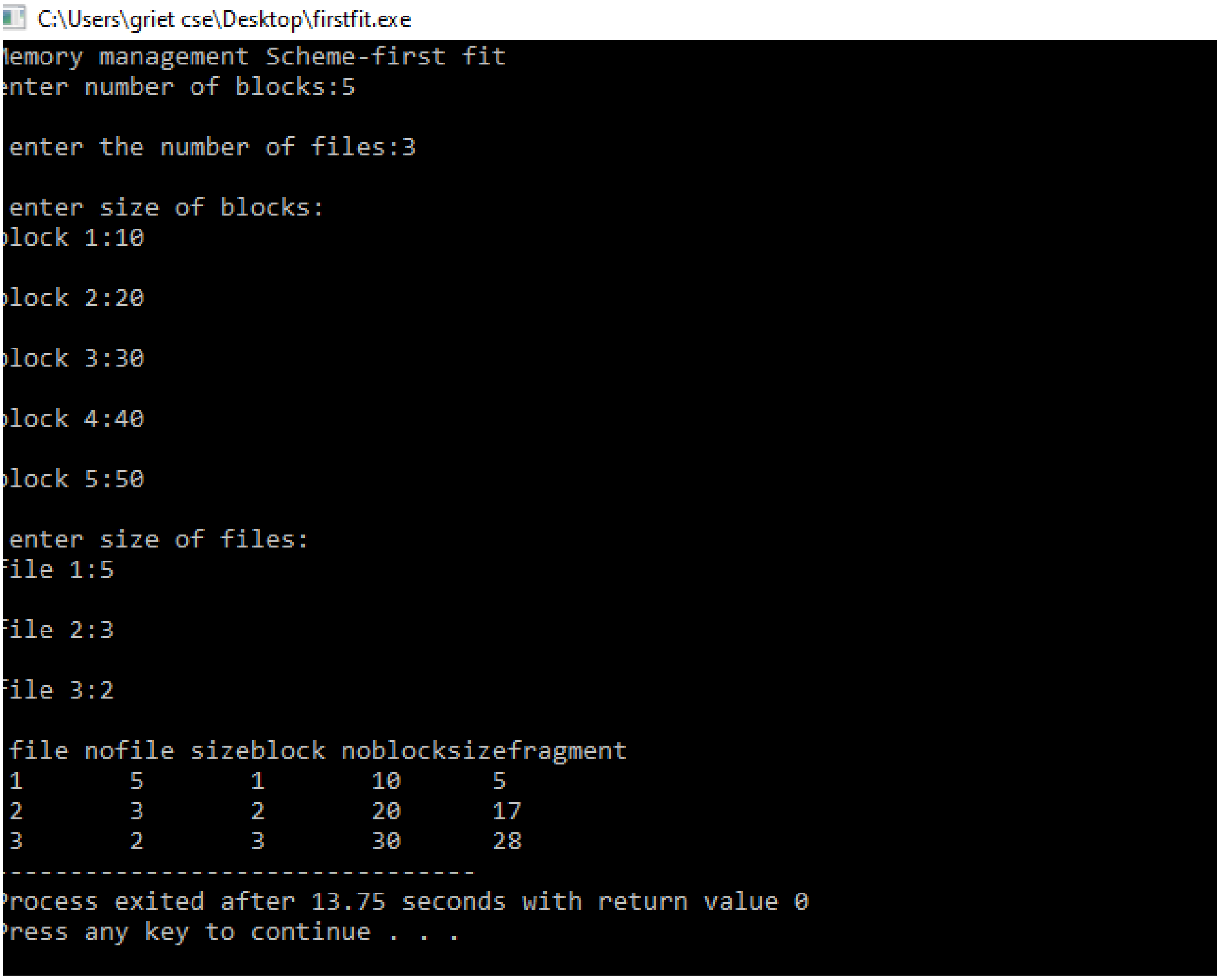
Task 10

Task 10a: To write C program to simulate First Fit Algorithm of Memory Management

Program:

```
#include<stdio.h>
#define max 25
void main()
{ int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
printf("Memory management Scheme-first fit");
printf("\nenter number of blocks:");
scanf("%d",&nb);
printf("\n enter the number of files:");
scanf("%d",&nf);
printf("\n enter size of blocks:");
for(i=1;i<=nb;i++)
   {     printf("\nblock %d:",i);
scanf("%d",&b[i]);
   }
printf("\n enter size of files:");
for(i=1;i<=nf;i++)
   {
printf("\nfile %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("\n file no\tfile size\tblock no\tblocksize\tfragment");
for(i=1;i<=nf;i++)
printf("\n %d \t %d \t %d \t %d \t %d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

Output:



C:\Users\griet cse\Desktop\firstfit.exe

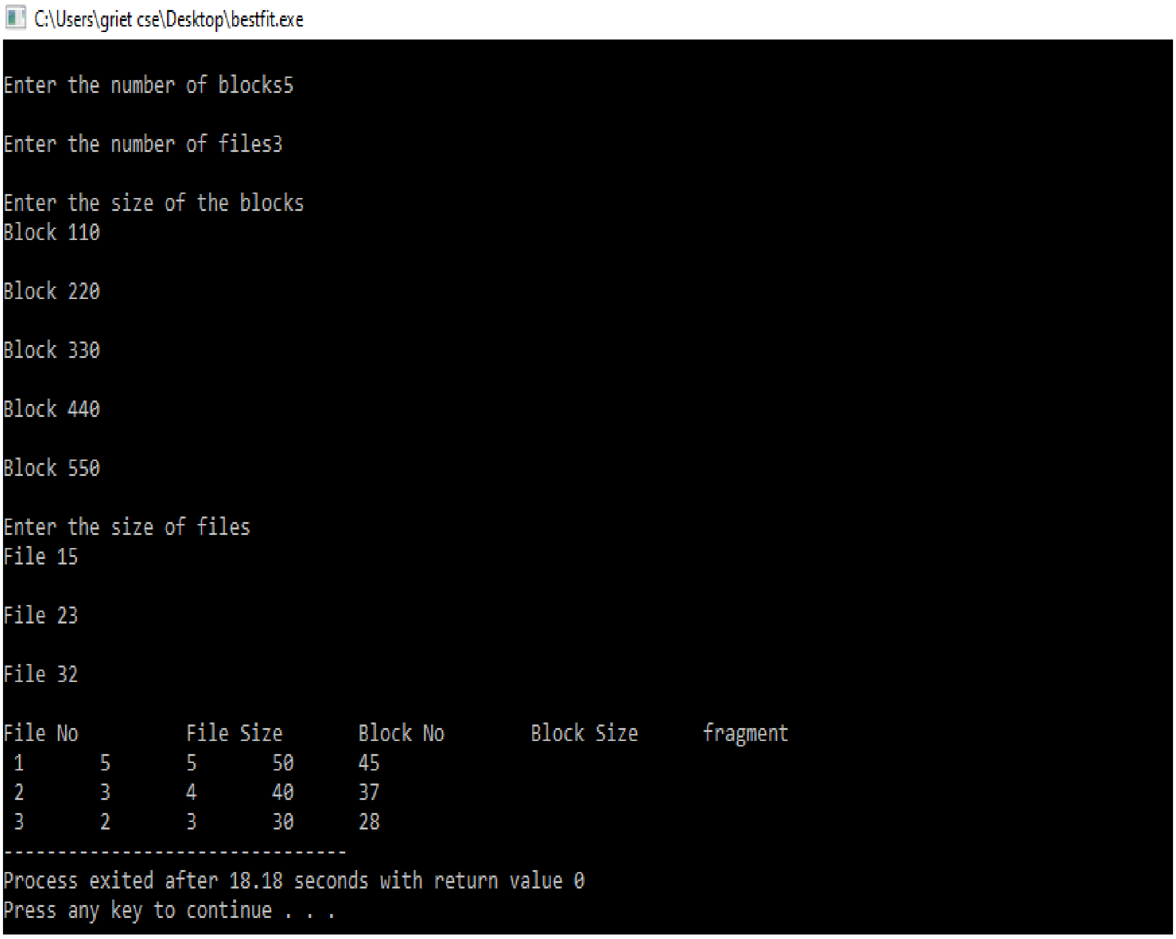
```
lemory management Scheme-first fit
nter number of blocks:5

 enter the number of files:3

 enter size of blocks:
lock 1:10

lock 2:20

lock 3:30

lock 4:40

lock 5:50

 enter size of files:
ile 1:5

ile 2:3

ile 3:2

 file nofile sizeblock noblocksizefragment
 1       5       1       10      5
 2       3       2       20      17
 3       2       3       30      28
-------------------------------
rocess exited after 13.75 seconds with return value 0
ress any key to continue . . .
```

Task 10b: To write C program to simulate Best Fit Algorithm of Memory Management

Program:

```
#include<stdio.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("\nEnter the number of blocks");
scanf("%d",&nb);
printf("\nEnter the number of files");
scanf("%d",&nf);
printf("\nEnter the size of the blocks");
for(i=1;i<=nb;i++)
{
printf("\nBlock %d",i);
scanf("%d",&b[i]);
}
printf("\nEnter the size of files");
for(i=1;i<=nf;i++)
{
printf("\nFile %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;
                                lowest=temp;
                                }
                        }
                }
        frag[i]=lowest;
        bf[ff[i]]=1;
        lowest=10000;
}

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

Output:



```

Enter the number of blocks5

Enter the number of files3

Enter the size of the blocks
Block 110

Block 220

Block 330

Block 440

Block 550

Enter the size of files
File 15

File 23

File 32

File No         File Size       Block No        Block Size      fragment
 1      5       5       50      45
 2      3       4       40      37
 3      2       3       30      28
--------------------------------
Process exited after 18.18 seconds with return value 0
Press any key to continue . . .
```

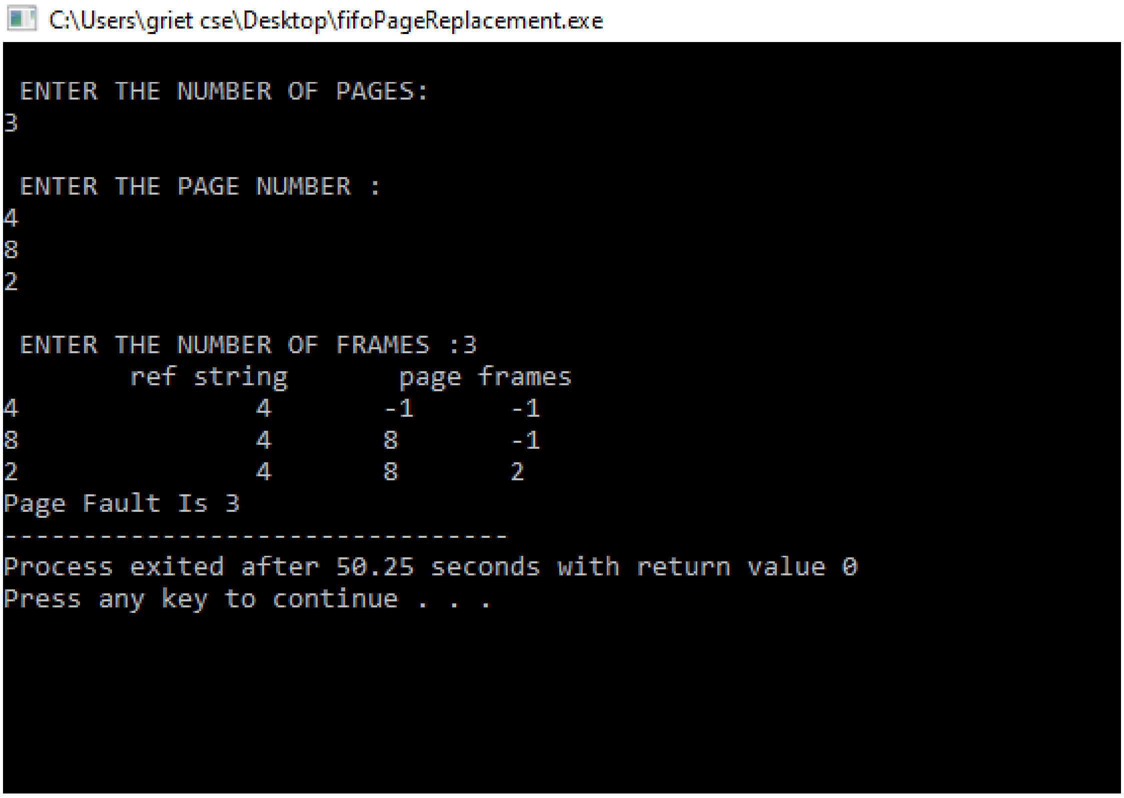
Task 11

Task 11a: To write C program to Simulate FIFO page replacement Algorithms formemory management:

Program:

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

Output:



```
 ENTER THE NUMBER OF PAGES:
3

 ENTER THE PAGE NUMBER :
4
8
2

 ENTER THE NUMBER OF FRAMES :3
        ref string       page frames
4               4       -1      -1
8               4       8       -1
2               4       8       2
Page Fault Is 3
--------------------------------
Process exited after 50.25 seconds with return value 0
Press any key to continue . . .
```

Task 11b: To write C program to Simulate LRU page replacement Algorithms for memory management:

Program:

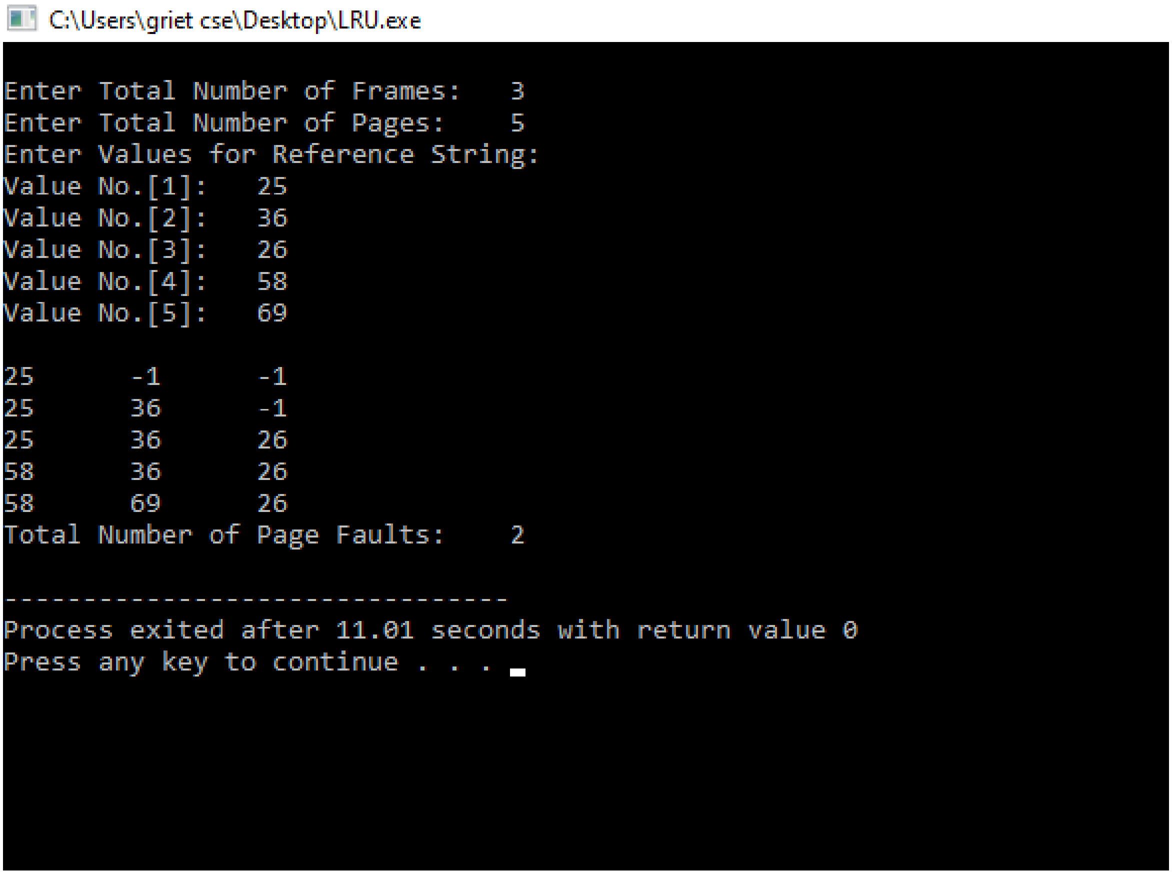
```
#include<stdio.h>

int main()

{

int frames[10], temp[10], pages[10];

int total_pages, m, n, position, k, l, total_frames;

int a = 0, b = 0, page_fault = 0;

printf("\nEnter Total Number of Frames:\t");

scanf("%d", &total_frames);

for(m = 0; m < total_frames; m++)

{

frames[m] = -1;

}

printf("Enter Total Number of Pages:\t");

scanf("%d", &total_pages);

printf("Enter Values for Reference String:\n");

for(m = 0; m < total_pages; m++)

{

printf("Value No.[%d]:\t", m + 1);

scanf("%d", &pages[m]);

}

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

{

a = 0, 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];

page_fault++;

}

printf("\n");

for(m = 0; m < total_frames; m++)

{

printf("%d\t", frames[m]);

}
```

```
}

printf("\nTotal Number of Page Faults:\t%d\n", page_fault);

return 0;

}
```

Output :



```
Enter Total Number of Frames:   3
Enter Total Number of Pages:    5
Enter Values for Reference String:
Value No.[1]:   25
Value No.[2]:   36
Value No.[3]:   26
Value No.[4]:   58
Value No.[5]:   69

25      -1      -1
25      36      -1
25      36      26
58      36      26
58      69      26
Total Number of Page Faults:    2

--------------------------------
Process exited after 11.01 seconds with return value 0
Press any key to continue . . .
```

Task 12

Task 12a: To write a C program to implement the concept of Indexing

Program:

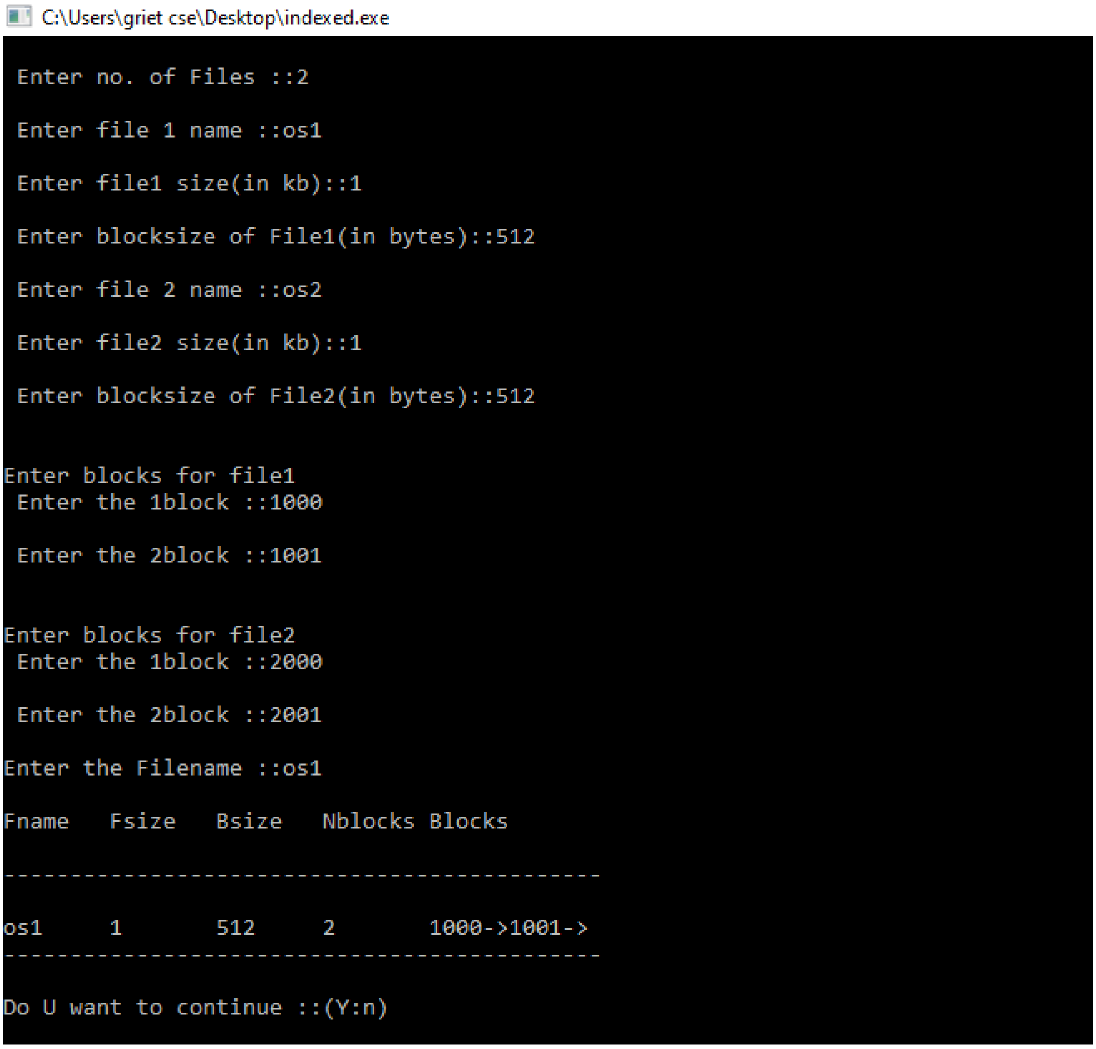
Indexed file allocation:

```
#include<stdio.h>

#include<string.h>

int n;

void main()

{

int b[20],b1[20],i,j,blocks[20][20],sz[20];

char F[20][20],S[20],ch;

int sb[20],eb[20],x;

printf("\n Enter no. of Files ::");

scanf("%d",&n);

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

{

printf("\n Enter file %d name ::",i+1);

scanf("%s",&F[i]);

printf("\n Enter file%d size(in kb)::",i+1);

scanf("%d",&sz[i]);

printf("\n Enter blocksize of File%d(in bytes)::",i+1);

scanf("%d",&b[i]);

}

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

{
```

```
b1[i]=(sz[i]*1024)/b[i];
printf("\n Enter Starting block of file%d::",i+1);
scanf("%d",&sb[i]);
printf("\n Enter Ending block of file%d::",i+1);
scanf("%d",&eb[i]);
printf("\nEnter blocks for file%d::\n",i+1);
for(j=0;j<b1[i]-2;)
{
printf("\n Enter the %dblock ::",j+1);
scanf("%d",&x);
if(x>sb[i]&&x<eb[i])
{
blocks[i][j]=x;
j++;
}
else
printf("\n Invalid block::");
}
}
do
{
printf("\nEnter the Filename ::");
scanf("%s",&S);
for(i=0;i<n;i++)
{
```

```
if(strcmp(F[i],S)==0)

{

printf("\nFname\tFsize\tBsize\tNblocks\tBlocks\n");

printf("\n------------------------------------------------------------\n");

printf("\n%s\t%d\t%d\t%d\t",F[i],sz[i],b[i],b1[i]);

printf("%d->",sb[i]);

for(j=0;j<b1[i]-2;j++)

printf("%d->",blocks[i][j]);

printf("%d->",eb[i]);

}

}

printf("\n------------------------------------------------------------\n");

printf("\nDo U want to continue (Y:n)::");

scanf("%d",&ch);

}while(ch!=0);

}
```

Output:



```
 Enter no. of Files ::2

 Enter file 1 name ::os1

 Enter file1 size(in kb)::1

 Enter blocksize of File1(in bytes)::512

 Enter file 2 name ::os2

 Enter file2 size(in kb)::1

 Enter blocksize of File2(in bytes)::512


Enter blocks for file1
 Enter the 1block ::1000

 Enter the 2block ::1001


Enter blocks for file2
 Enter the 1block ::2000

 Enter the 2block ::2001

Enter the Filename ::os1

Fname   Fsize   Bsize   Nblocks Blocks

---------------------------------------------

os1     1       512     2       1000->1001->
---------------------------------------------

Do U want to continue ::(Y:n)
```

Task 12a: To write a C program to implement the concept of Hashing.

Program:

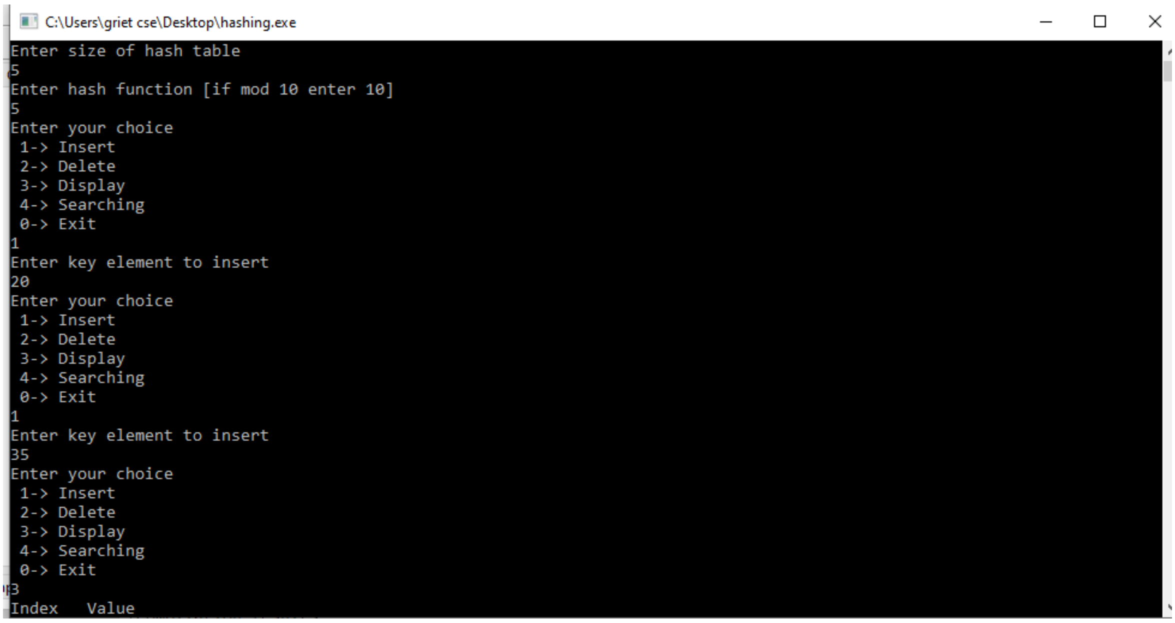
```
#include<stdio.h>
#include<limits.h>
 /*
This is code for linear probing in open addressing. If you want to do quadratic probing and
double hashing which are also
open addressing methods in this code when I used hash function that (pos+1)%hFn in that
place just replace with another function.
*/
 void insert(int ary[],int hFn, int size){
    int element,pos,n=0;
            printf("Enter key element to insert\n");
            scanf("%d",&element);
            pos = element%hFn;
            while(ary[pos]!= INT_MIN) { // INT_MIN and INT_MAX indicates that cell is
empty. So if cell is empty loop will break and goto bottom of the loop to insert element
            if(ary[pos]== INT_MAX)
            break;
            pos = (pos+1)%hFn;
            n++;
            if(n==size)
            break;      // If table is full we should break, if not check this, loop will go to infinite
loop.
            }
            if(n==size)
        printf("Hash table was full of elements\nNo Place to insert this element\n\n");
            else
```

```
     ary[pos] = element;    //Inserting element
}
void delet(int ary[],int hFn,int size){
        /*
        very careful observation required while deleting. To understand code of this delete
function see the note at end of the program
        */
        int element,n=0,pos;
        printf("Enter element to delete\n");
        scanf("%d",&element);
        pos = element%hFn;
        while(n++ != size){
        if(ary[pos]==INT_MIN){
                printf("Element not found in hash table\n");
                break;
        }
        else if(ary[pos]==element){
                ary[pos]=INT_MAX;
                printf("Element deleted\n\n");
                break;
        }
        else{
        pos = (pos+1) % hFn;
        }
        }
        if(--n==size)
     printf("Element not found in hash table\n");
```

```
   }
void search(int ary[],int hFn,int size){
    int element,pos,n=0;
    printf("Enter element you want to search\n");
    scanf("%d",&element);
    pos = element%hFn;
    while(n++ != size){
    if(ary[pos]==element){
        printf("Element found at index %d\n",pos);
        break;
    }
    else
    if(ary[pos]==INT_MAX ||ary[pos]!=INT_MIN)
      pos = (pos+1) %hFn;
    }
    if(--n==size) printf("Element not found in hash table\n");
}
void display(int ary[],int size){
    int i;

    printf("Index\tValue\n");

    for(i=0;i<size;i++)
   printf("%d\t%d\n",i,ary[i]);
}
int main(){
    int size,hFn,i,choice;
    printf("Enter size of hash table\n");
    scanf("%d",&size);

    int ary[size];

    printf("Enter hash function [if mod 10 enter 10]\n");
    scanf("%d",&hFn);

    for(i=0;i<size;i++)
ary[i]=INT_MIN; //Assigning INT_MIN indicates that cell is empty

    do{
    printf("Enter your choice\n");
    printf(" 1-> Insert\n 2-> Delete\n 3-> Display\n 4-> Searching\n 0-> Exit\n");
    scanf("%d",&choice);

    switch(choice){
            case 1:
                    insert(ary,hFn,size);
                    break;
            case 2:
                    delet(ary,hFn,size);
                    break;
            case 3:
                    display(ary,size);
                    break;
```

```
                case 4:
                        search(ary,hFn,size);
                        break;
                default:
                        printf("Enter correct choice\n");
                        break;
        }
        }while(choice);

        return 0;
}
```

Output:



```
Enter size of hash table
5
Enter hash function [if mod 10 enter 10]
5
Enter your choice
 1-> Insert
 2-> Delete
 3-> Display
 4-> Searching
 0-> Exit
1
Enter key element to insert
20
Enter your choice
 1-> Insert
 2-> Delete
 3-> Display
 4-> Searching
 0-> Exit
1
Enter key element to insert
35
Enter your choice
 1-> Insert
 2-> Delete
 3-> Display
 4-> Searching
 0-> Exit
3
Index   Value
```