EXP NO : 1 **DATE :**

UNIX INTRODUCTION

AIM :

<u>UNIX Commands</u>

<u>General Commands</u>:

1**. date**: This tells the current date and time.

$ date

Thu Oct 15 09:34:50 PST 2005

2**. who**: Gives the details of the user who have logged into the system.

$ who

abc tty() oct 15 11:17

Xyz tty4 oct 15 11:30

3. **whoami**: Gives the details regarding the login time and system's name for the connection being used.

$ whoami.

user 1 ttya Oct 15 12:20.

4. **man**: It displays the manual page of our terminal with the command 'man' command name.

$ man who

5. **head and tail**: 'head' is used to display the initial part of the text file and 'tail' is used to display the last part of the file.

$ head [-count] [filename]

$ tail [-count] [filename]

6. **pwd**: It displays the full path name for the current directory we are working in.

$ pwd /usr/tmp.

7. is : It displays the list of files in a current working directory.

$ ls $ ls -l = lists files in long format.

$ ls –t = lists in order of last modification time.

$ ls –a = lists all entries , including the hidden files.

$ ls –d = lists directory files instead of its contents.

$ ls –p = puts a slash after each directory.

$ ls –u = lists in order of last access time.

8. **mkdir**: It is used to create a new directory.

9. **cd** : It is used to change from the working directory to any other directory specified.

$ cd $ cd.. $ cd / $ cd dirl changes to home directory. changes to parent directory. changes to root directory. changes to directory dirl.

10**. rmdir**: It is use to remove the directory specified in the command line.

$ rmdir directory name

11. **cat** : This command helps us to specify the contents of the file we specify.

$ cat [option....[file....]]

12**. cp** : This command is used to create duplicate copied of ordinary files.

$ cp file target

$ cp file1 file2 [file1 is copied to file2]

13. **mv**: This command is used to rename and move ordinary and directory files.

$ mv file1 file2

$ mv directory1 directory2

14. **ln**: This is used to link file.

$ ln file1 [file2....] target

15. **rm**: This command is used to remove one or more files from the directory.This can be used to delete all files as well as directory.

$ rm [option.....] file

16. **chmod**: Change the access permissions of a file or a directory.

$ chmod mode file

$ chmod [who] [+/-/=] [permission...] file

who = a – all users g

– group o

– others u –user

[+/-/=] + adds

- removes

= assigns

[permission] r = read

w =write

x =execute Ex.::

$ chmod 754 prog1.

17. **Chown** : change the owner ID of the files or directories.

Owner may be decimal user ID or a login name found in the file/etc/passwd. This utility is governed by the chown kernel authorization. If it is not granted, ownership can only be changed by root.

Ex:: $ chown tutor test

18**. wc**: counts and displays the lines, words and characters in the files specified.

$ wc

$ wc

$ wc

$wc

Ex:: $ wc prog2.

3 9 60 prog2.

19. **grep** : searches the file for the pattern.

$ grep [option...] pattern [file....].

Display the lines containing the pattern on the standard output.

$ grep c report only the number of matching lines.

$ grep-l list only the names of files containing pattern.

$grep-v display all lines expert those containing pattern.

Ex: grep c”the”prog2.

20. **cut**: cuts out selected fields of each line of a file.

$ cut -first [-d char] [file1 file2...]. -d = it is delimiter. Default is tab.

Ex: cut –f1, 3 –d”w”prog2.

21. **paste**: merges the corresponding lines of the given files.

$ paste –d file1 file2

Option – d allows replacing tab character by one or more alternate characters.

Ex:: paste prog1 prog2

22. **sort** : arranges lines in alphabetic or numeric order.

$ sort [option]file.

Option –d dictionary order

Option –n arithmetic order

Option –r reverse order

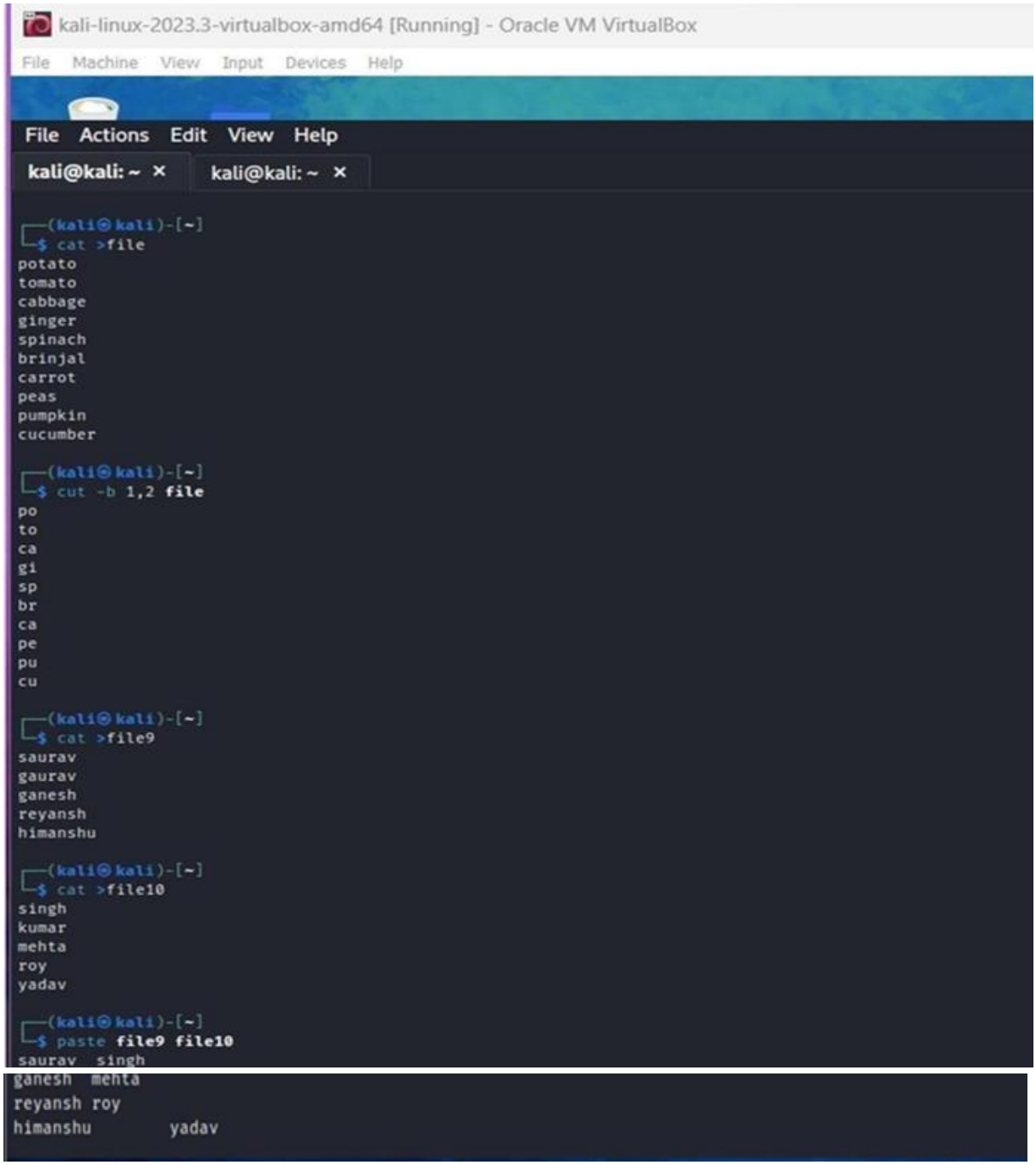
Ex :: $ ls – l | sort –n

OUTPUT :

```
File Machine View Input Devices Help

File Actions Edit View Help

┌──(kali㉿kali)-[~]
└─$ date
Sun Sep 17 06:20:30 AM EDT 2023

┌──(kali㉿kali)-[~]
└─$ who
kali     tty7         2023-09-17 04:48 (:0)

┌──(kali㉿kali)-[~]
└─$ whoami
kali

┌──(kali㉿kali)-[~]
└─$ man who

┌──(kali㉿kali)-[~]
└─$ vi file1.txt

┌──(kali㉿kali)-[~]
└─$ cat file1.txt
Hii ...
I am a student.
I studied in Dr.Mgr University Chennai.


┌──(kali㉿kali)-[~]
└─$ pwd
/home/kali

┌──(kali㉿kali)-[~]
└─$ ls
1126    banker.c  dining.c   fibonacci.sh        fibo.sh    first.cpp  gk.sh      Pictures          robin.cpp     singh2     worst.c
add.sh  best.c    Documents  fibonacci.sh.save   fib.sh     gaurav     greater.sh positive.sh       shortest.cpp  system.cpp
a.out   cpu.c     Downloads  fibonacci.sh.save.1 file1.txt  gaurav1    memory.c   positive.sh.save  singh         Templates
arm.sh  Desktop   fact.sh    fibonacci.sh.save.2 file2      gaurav2    Music      Public            singh1        Videos

┌──(kali㉿kali)-[~]
└─$ mkdir file3

┌──(kali㉿kali)-[~]
└─$ cd Desktop/

┌──(kali㉿kali)-[~/Desktop]
└─$ cd

┌──(kali㉿kali)-[~]
└─$ rmdir file3
```

```
kali-linux-2023.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File  Machine  View  Input  Devices  Help
File  Actions  Edit  View  Help
kali@kali: ~/Downloads ×   kali@kali: ~ ×
┌──(kali㉿kali)-[~]
└─$ cat file1.txt

┌──(kali㉿kali)-[~]
└─$ cat file2.txt

┌──(kali㉿kali)-[~]
└─$ cp file1.txt dir1

┌──(kali㉿kali)-[~]
└─$ mkdir file4

┌──(kali㉿kali)-[~]
└─$ cp file1.txt file2.txt file4

┌──(kali㉿kali)-[~]
└─$ mv file1.txt file2.txt

┌──(kali㉿kali)-[~]
└─$ ln -s file1

┌──(kali㉿kali)-[~]
└─$ ls -l file1
lrwxrwxrwx 1 kali kali 5 Sep 17 06:31 file1 → file1

┌──(kali㉿kali)-[~]
└─$ cat >file1.txt
I am a student.

┌──(kali㉿kali)-[~]
└─$ cat >file2.txt
I am an Engineer.

┌──(kali㉿kali)-[~]
└─$ rm file1.txt file2.txt

┌──(kali㉿kali)-[~]
└─$ cd Downloads/

┌──(kali㉿kali)-[~/Downloads]
└─$ mkdir file1

┌──(kali㉿kali)-[~/Downloads]
└─$ mkdir file2
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ chmod u+r file5.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ touch testfile2

┌──(kali㉿kali)-[~/Downloads]
└─$ ls -l testfile2
-rw-r--r-- 1 kali kali 0 Sep 17 06:42 testfile2

┌──(kali㉿kali)-[~/Downloads]
└─$ cd

┌──(kali㉿kali)-[~]
└─$ wc positive.sh
 12  51 212 positive.sh

┌──(kali㉿kali)-[~]
└─$ grep file1
Temple Name:
Kedarnath
Vaishno Devi
Jagannath Temple
Tirupati Balaji
Mukteswara
Siddhivinayak
Badrinath
```


kali-linux-2023.3-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
kali@kali: ~
kali@kali: ~
(kali㉿kali)-[~]
$ cat >file
potato
tomato
cabbage
ginger
spinach
brinjal
carrot
peas
pumpkin
cucumber
(kali㉿kali)-[~]
$ cut -b 1,2 file
po
to
ca
gi
sp
br
ca
pe
pu
cu
(kali㉿kali)-[~]
$ cat >file9
saurav
gaurav
ganesh
reyansh
himanshu
(kali㉿kali)-[~]
$ cat >file10
singh
kumar
mehta
roy
yadav
(kali㉿kali)-[~]
$ paste file9 file10
saurav singh
ganesh mehta
reyansh roy
himanshu yadav

RESULT :

EXP NO : 2 **DATE :**

SHELL PROGRAMING

AIM:

SHELL PROGRAMS

a. SUM OF TWO NUMBERS

Aim:

Algorithm

Program

```
echo " enter first no " read a

echo " enter second no " read b

c= expr $a + $b
```

INPUT

[student@localhost student] $sh sum

Enter first no 10

enter second no 20

OUTPUT

```
enter first no
10
enter second no
20
30
```

b. FIBONACCI SERIES

Aim:

Algorithm

Program:

echo "enter a number" read n

i=0

```
a=0
b=1
c=0
echo "fibonacci series is"
echo "$a"
echo "$b"
n1=` expr $n - 2`
while [ $i -lt $n1 ]
do
c=` expr $a + $b`
echo "$c"
a=$b
b=$c
i=` expr $i + 1`
done
```

INPUT:

```
[student@localhost student]$ sh fibo
enter a number
7
```

OUTPUT:

```
administrator@administrator-Veriton-Series:~/operating$ sh fib.sh
enter a number
7
fibonacci series is
0
1
1
2
3
5
8
```

c. POSITIVE OR NEGATIVE

Aim:

Algorithm

Program:

```
echo " enter a number " read a

if [ $a -eq 0 ]

then

echo " no is zero " elif [ $a -gt 0

] then

echo " no is positive " else

echo " no is negative "fi
```

INPUT :

[student@localhost student]$ sh positive

Enter a number : 10

OUTPUT:

```
administrator@administrator-Veriton-Series:~/operating$ sh pos.sh
enter a number
10
no is postive
```

d. GREATEST OF THREE NUMBERS

Aim:

Algorithm

Program:

```
echo " enter the three numbers " read x
read y
read z
echo " enter the three numbers " read x
read y
read z
if [ $x -gt $y -a $x -gt $z ] then
echo "$x is greater"
elif [ $y -gt $x -a $y -gt $z ] then
echo "$y is greater" else
echo "$z is greater"
fi
```

INPUT:

[student@localhost student]$ sh greatest enter the three

numbers

12

13

45

OUTPUT:

```
administrator@administrator-Veriton-Series:~/operating$ sh gre.sh
enter the three numbers
12
13
45
45 is greater
```

e. ARMSTRONG NUMBER

Aim:

Algorithm

Program:

```
echo "enter the number" read num

ans=0

n=$num


while [ $n -gt 0 ] do

q=` expr $n % 10`

ans=` expr $ans + $q \* $q \* $q` n=` expr $n / 10`

done

if [ $ans -eq $num ] then

echo "number is armstrong" else

echo "number is not armstrong"fi
```

INPUT:

[student@localhost student]$ sh armstrong enter the

number

153

OUTPUT:

number is Armstrong

INPUT:

[student@localhost student]$ sh armstrong enter the

number

205

OUTPUT:

```
administrator@administrator-Veriton-Series:~/operating$ sh arm.sh
enter the number
205
number is not armstrong
```

f. FACTORIAL NUMBER

Aim :

Algorithm :

Program:

```
n=0
fact=1
g=0
y=1
echo "enter no to find the factorial:"read n
g=$n
while [ $n -ge $y ]
do
fact=` expr $fact \* $n
` n=` expr $n - 1 `
done
echo “factorial for $g is fact”
```

INPUT:

[student@localhost student]$sh factenter no to find the factorial :

8

OUTPUT:

```
administrator@administrator-Veriton-Series:~/operating$ sh fac.sh
enter number to find the factorial
8
factorial for 8 is 40320
```

RESULT :

EXP NO : 3 **DATE :**

PROCESS CREATION

AIM:

ALGORITHM:

PROGRAM :

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

pid_ t p = fork();

if (p<0) {

}

perror("fork fail");

exit(1);

printf("Hello world!, process_id (pid) = %d\n", getpid());

return 0;
```

INPUT:

```
[student@localhost student]$ cc proc.c

[student@localhost student]$ ./a.out
```

OUTPUT:

```
telnet@telnet-HP-Compaq-Pro-6300-SFF:~/srm$ ./a.out
Hello world!,process_id(pid)=2025549
telnet@telnet-HP-Compaq-Pro-6300-SFF:~/srm$ Hello world!,process_id(pid)=2025550
```

RESULT :

EXP NO: 4 **DATE :**

INTERPROCESS COMMUNICATION USING SHARED MEMORY

AIM:

ALGORITHM:

PROGRAM :

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/ipc.h>

#include <sys/shm.h>

 #define SHM_SIZE 1024

 int main() {

   key_t key = ftok("shmfile", 65);

    int shmid = shmget(key, SHM_SIZE, IPC_CREAT | 0666);

   if (shmid == -1) {

      perror("shmget");

      exit(1);

   }

  char *shmaddr = (char*)shmat(shmid, NULL, 0);

   if (shmaddr == (char*)-1) {

      perror("shmat");

      exit(1);

   }

strcpy(shmaddr, "hai, the message is shared!");

  shmdt(shmaddr);

   shmaddr = (char*)shmat(shmid, NULL, 0);

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

  shmdt(shmaddr);

 shmctl(shmid, IPC_RMID, NULL);

   return 0;

}
```

OUTPUT :

```
telnet@telnet-HP-Compaq-Pro-6300-SFF:~/srm$ ./a.out
Hello world!,process_id(pid)=2025549
telnet@telnet-HP-Compaq-Pro-6300-SFF:~/srm$ Hello world!,process_id(pid)=2025550
```

RESULT :

EXP NO : 5 **DATE :**

CPU SCHEDULING ALGORITHMS

AIM :

ALGORITHM :

EXP NO : 5A **DATE :**

FIRST COME FIRST SERVED CPU SCHEDULING

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
main()
{
 int i,n,w[10],e[10],b[10]; float wa=0,ea=0;
printf("\nEnter the no of jobs: ");
scanf("%d",&n); for(i=0;i<n;i++)
{
 printf("\n Enter the burst time of job %d :",i+1);
scanf("%d",&b[i]);
if(i==0)

{
w[0]=0;
e[0]=b[0];
 }
 else
 {
 e[i]=e[i-1]+b[i];
 w[i]=e[i-1];
 }
 }
 printf("\n\n\tJobs\tWaiting time \tBursttime\tExecution time\n");
printf("\t_______________ \n"); for(i=0;i<n;i++)
{
printf("\t%d\t\t%d\t\t%d\t\t%d\n",i+1,w[i],b[i],e[i]); wa+=w[i];
 ea+=e[i];
 }
```

```
 wa=wa/n; ea=ea/n;
 printf("\n\nAverage waiting time is :%2.2f ms\n",wa);
printf("\nAverage execution time is:%2.2f ms\n\n",ea);
}
```

INPUT :

Enter the no of jobs: 4

Enter the burst time of job 1 : 5

Enter the burst time of job 2 : 4

Enter the burst time of job 3 : 3

Enter the burst time of job 4 : 6

OUTPUT :

```
Enter the no of jobs: 4

 Enter the burst time of job 1 :5

 Enter the burst time of job 2 :4

 Enter the burst time of job 3 :3

 Enter the burst time of job 4 :6

        Jobs    Waiting time    Bursttime       Execution time
        -------------------------------------------------------
        1               0               5               5
        2               5               4               9
        3               9               3               12
        4               12              6               18

Average waiting time is :6.50 ms

Average execution time is:11.00 ms

-
```

RESULT :

EXP NO : 5B **DATE :**

SHORTEST JOB FIRST CPU SCHEDULING

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
struct sjfs
{
char pname[10];
int btime;
}
proc[10],a;
void main( )
{
struct sjfs proc[10];
int n,i,j;
int temp=0,temp1=0,temp2;
char name[20];
float tt,awt;
printf("Enter the number of processes:\n");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("Enter the process name: \n");
scanf("%s",&proc[i].pname);
printf("Enter the Burst time:\n");
scanf("%d",&proc[i].btime);
}
for(i=0;i<n;i++)
{
printf("Enter the process name: \n");
scanf("%s",&proc[i].pname);
printf("Enter the Burst time:\n");
```

```
scanf("%d",&proc[i].btime);
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(proc[i].btime<proc[j].btime)
{
a=proc[i];
proc[i]=proc[j];
proc[j]=a;
}
}
}
printf("-------------------- CPU SCHEDULING ALGORITHM - SJFS ----------------------------");
printf("\n\tprocess name \tBurst time \twaiting time \tturnaround time\n");
temp=0;
for(i=0;i<n;i++)
{
temp=temp1+temp+proc[i].btime;
temp1=temp1+proc[i].btime;temp2=temp1- proc[i].btime;
printf("\n\t %s\t %d ms\t %d ms \t %d ms\n",proc[i].pname,proc[i].btime,temp2,temp1);
}
printf("------------------------------------------------------------ ");
awt=(temp-temp1)/n;
tt=temp/n;
printf("\nThe Average Waiting time is %4.2f milliseconds\n",awt);
printf("\nThe Average Turnaround time is %4.2f",tt);
}
```

INPUT:

Enter the number of processes: 4

Enter the process name: p1

Enter the Burst time: 5

Enter the process name: p2

Enter the Burst time: 5

Enter the process name: p3

Enter the Burst time: 6

Enter the process name: p4

Enter the Burst time: 2

OUTPUT :

```
Enter the number of processes:
4
Enter the process name:
p1
Enter the Burst time:
5
Enter the process name:
p2
Enter the Burst time:
5
Enter the process name:
p3
Enter the Burst time:
6
Enter the process name:
p4
Enter the Burst time:
2
-------------------- CPU SCHEDULING ALGORITHM - SJFS --------------------

        process name    Burst time      waiting time    turnaround time

         p4      2 ms    0 ms    2 ms

         p1      5 ms    2 ms    7 ms

         p2      5 ms    7 ms    12 ms

         p3      6 ms    12 ms   18 ms
-----------------------------------------------------------------------
The Average Waiting time is 5.00 milliseconds

The Average Turnaround time is 9.00
```

RESULT :

EXP NO : 5C **DATE :**

ROUND ROBIN CPU SCHEDULING

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
#include<malloc.h>
void line(int i)
{
 int j;
 for(j=1; j<=i; j++)
  printf("-");
 printf("\n");
}
struct process
{
 int p_id;
 int etime, wtime, tatime;
 };
 struct process *p, *tmp;
 int i, j, k, l, n, time_slice, ctime;
 float awtime=0, atatime=0;
 int main()
{
 printf("Process Scheduling - Round Robin \n");
 line(29);
 printf("Enter the no.of processes : ");
 scanf("%d", &n);
 printf("Enter the time slice : ");
 scanf("%d", &time_slice);
 printf("Enter the context switch time : ");
 scanf("%d", &ctime);
 p=(struct process*) calloc(n+1, sizeof(struct process));
```

```
 tmp=(struct process*) calloc(n+1, sizeof(struct process));
 for(i=1; i<=n; i++)
 {
   printf("Enter the execution time of process %d : ", i-1);
   scanf("%d", &p[i].etime);
   p[i].wtime=(time_slice+ctime)*i-1;
   awtime += p[i].wtime;
   p[i].p_id=i-1;
   tmp[i]=p[i];
 }
 i=0; j=1; k=0;
while(i<n)
 {
   for(j=1; j<=n; j++)
   {
         if(tmp[j].etime <= time_slice && tmp[j].etime!=0)
     {
        k=k+tmp[j].etime;
        tmp[j].etime=0;
        p[j].tatime=k;
        atatime += p[j].tatime;
        k=k+ctime;
        i++;
     }
     if(tmp[j].etime>time_slice && tmp[j].etime!=0)
     {
        k=k+time_slice+ctime;
        tmp[j].etime -= time_slice;
     }
   }
```

```
 }
 awtime=awtime/n;
 atatime=atatime/n;
 printf("\nShedule \n");
 line(60);
 printf("Process\t\tExecution\tWait\t\tTurnaround\n");
 printf("Id No\t\ttime\t\ttime\t\ttime\n");
 line(60);
 for(i=1;i<=n;i++)
     printf("%7d\t%14d\t%8d\t%14d \n", p[i].p_id, p[i].etime, p[i].wtime,
 p[i].tatime);
 line(60);
 printf("Avg waiting time     :\t%2f \n",awtime);
 printf("Avg turn around time :\t%2f\n",atatime);
 line(60);
 return(0);
}
```

Input :

```
Round Robin Scheduling -------------------------------------------------------
Enter the no of processes : 3
Enter the time slice : 4
Enter the context switch time : 5
Enter the execution time of process 0 : 6
Enter the execution time of process 1 : 6
Enter the execution time of process 2 : 5
```

OUTPUT :

```
Process Scheduling - Round Robin
Enter the no.of processes : 3
Enter the time slice : 4
Enter the context switch time : 5
Enter the execution time of process 0 : 6
Enter the execution time of process 1 : 6
Enter the execution time of process 2 : 5

Shedule
-------------------------------------------------------------
Process        Execution      Wait           Turnaround
Id No          time           time           time
-------------------------------------------------------------
     0            6            8                29
     1            6           17                36
     2            5           26                42
-------------------------------------------------------------
Avg waiting time :      17.000000
Avg turn around time :  35.666668
-------------------------------------------------------------
```

RESULT :

EXP NO : 5D **DATE :**

PRIORITY CPU SCHEDULING

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
#include<malloc.h>
void line(int i)
{
 int j; for(j=1;j<=i;j++)
printf("-");
 printf("\n");
 }
 struct process
 {
 int p_id,priority;
 int etime,wtime,tatime;
 };
 struct process *p,temp;int i,j,k,l,n;
float awtime=0,atatime=0; int main()
{
 printf("Priority Scheduling\n"); line(29);
 printf("Enter the no of processes : "); scanf("%d",&n);
 p=(struct process *) calloc(n+1,sizeof(struct process)); p[0].wtime=0;
p[0].tatime=0; for(i=1;i<=n;i++)
{
printf("Enter the execution time of process %d:",i-1); scanf("%d",&p[i].etime);
printf("Enter the priority of process %d:",i-1); scanf("%d",&p[i].priority);
p[i].p_id=i;
}
 for(i=1;i<=n;i++) for(j=1;j<=n-i;j++) if(p[i].priority>p[j+1].priority)
 {
 temp=p[j]; p[j]=p[j+1]; p[j+1]=temp;
```

```
}
 for(i=1;i<=n;i++)
 {
 p[i].wtime=p[i-1].tatime; p[i].tatime=p[i-1].tatime+p[i].etime;
awtime+=p[i].wtime; atatime+=p[i].tatime;
}
 awtime=awtime/n; awtime=atatime/n;
printf("\nSchedule\n");
printf("Process\t\texection\twait\t\tturnaround\n");
printf("Id No\t\t time\t\ttime\t\ttime\n");
for(i=1;i<=n;i++)
printf("%7d\t%14d\t%8d\t%14d\n",p[i].p_id,p[i].etime,p[i].tatime);
printf("Avg waiting time:\t %2f\n",awtime);
printf("Avg turnaround time:\t %2f\n",atatime); line(60);
return(0);
}
```

INPUT:

```
 Enter the no of processes : 3
 Enter the execution time of process0 : 5
Enter the priority of process0 : 6
Enter the execution time of process1: 3
Enter the priority of process1: 5
Enter the execution time of processes2: 1
Enter the priority of process2: 3
```

OUTPUT :

```
Priority Scheduling
---------------------------
Enter the no of processes : 3
Enter the execution time of process 0:5
Enter the priority of process 0:6
Enter the execution time of process 1:3
Enter the priority of process 1:5
Enter the execution time of process 2:1
Enter the priority of process 2:3

Schedule
------------------------------------------------------------
Process          exection        wait            turnaround
Id No             time           time            time
------------------------------------------------------------
      2             3             3                  3
      3             1             4                  3
      1             5             9                  3
------------------------------------------------------------
Avg waiting time:        5.333333
Avg turnaround time:     16.000000
------------------------------------------------------------
```

RESULT :

EXP NO : 6 **DATE :**

PRODUCER CONSUMER PROBLEM USING SHARED MEMORY

AIM :

ALGORITHM :

PROGRAM :

```
#include <stdio.h>

#define BUFFER_SIZE 5

int buffer[BUFFER_SIZE];

int in = 0;

int out = 0;

int count = 0;

void producer()

{

int x;

printf("\nEnter the Item to be produced:");

scanf("%d",&x);

if (count == BUFFER_SIZE)

printf("\nThe buffer is full");

else

{

buffer[in] = x;

in = (in + 1) % BUFFER_SIZE;

count++;

printf("\nThe next item produced is %d",x);

}}

void consumer()

{

int y;

if (count == 0)

{

printf("No item to consume\n");

}
```

```
else
{
y=buffer[out];
printf("The consumed item is %d",y);
out = (out + 1) % BUFFER_SIZE;
count--;
}
}
void main()
{
int ch;
char g='y';
do
{
printf("\n main Menu");
printf("\n 1.Produce \n 2.Consume \n 3.Exit \n");
printf("\n Enter your Choice");
scanf("%d", &ch);
switch(ch)
{
case 1:
producer();
break;
case 2:
consumer();
break;
case 3:
exit(1);
break;
default:
```

```
printf("\n Enter the correct choice:");
}
printf("\n Do u want to continue: ");
scanf("\n%c", &g);
} while(g=='y'||g=='Y');
}
```

INPUT:

```
[student@localhost student]$ gcc producer.c
[student@localhost student]$ ./a.out
```

OUTPUT :

```
telnet@telnet-HP-Compaq-Pro-6300-SFF:~/srm$ ./a.out

 main Menu
 1.Produce
 2.Consume
 3.Exit

 Enter your Choice 1

Enter the Item to be produced:34

The next item produced is 34
 Do u want to continue:  y

 main Menu
 1.Produce
 2.Consume
 3.Exit

 Enter your Choice 2
The consumed item is 34
 Do u want to continue: y

 main Menu
 1.Produce
 2.Consume
 3.Exit

 Enter your Choice 2
No item to consume

 Do u want to continue: y

 main Menu
 1.Produce
 2.Consume
 3.Exit

 Enter your Choice 1

Enter the Item to be produced: 55

The next item produced is 55
```

RESULT :

EXP NO : 7 **DATE :**

IMPLEMENTAION OF DINING PHILOSOPHER'S PROBLEM

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
char state[10],self[10],spoon[10];
void test(int k)
{
if((state[(k+4)%5]!='e')&&(state[k]=='h')&&(state[(k+1)%5]!='e'))
{
state[k]='e';
self[k]='s';
spoon[k]='n';
spoon[(k+4)%5]='n';
}
}
void pickup(int i)
{
state[i]='h';
test(i);
if(state[i]=='h')
{
self[i]='w';
}
}
void putdown(int i)
{
state[i]='t';
spoon[i]='s';
spoon[i-1]='s';
test((i+4)%5);
test((i+1)%5);
```

```
}
int main()
{
int ch,a,n,i;
printf("\t\t Dining Philosopher Problem\n");
for(i=0;i<5;i++)
{
state[i]='t';
self[i]='s';
spoon[i]='s';
}
printf("\t\t Initial State of Each Philosopher\n");
printf("\n\t Philosopher No.\t Think/Eat  \tStatus    \tspoon");
for(i=0;i<5;i++)
{
}
printf("\n\t\t %d\t\t%c\t\t%c\t%c\n",i+1,state[i],self[i],spoon[i]);
printf("\n 1.Exit \n 2.Hungry\n3.Thinking\n");
printf("\n Enter your choice\n");
printf("\t\t");
scanf("%d",&ch);
while(ch!=1)
{
switch(ch)
{
case 2:
{
printf("\n\t Enter which philosopher is hungry\n");
printf("\t\t");
scanf("%d",&n);
```

```
n=n-1;
pickup(n);
break;
}
case 3:
{
    printf("\n\t Enter which philosopher is thinking\n");
    printf("\t\t");
    scanf("%d",&n);
    n=n-1;
    putdown(n);
    break;
    }
}
printf("\n\t State of Each philosopher\n\n");
printf("\n\t Philosoper No.\t Thinking\t Hungry");
for(i=0;i<5;i++)
{
    printf("\n\t\t %d\t\t%c\t\t%c\t%c\n",i+1,state[i],self[i],spoon[i]);
}
printf("\n 1.Exit\n 2.Hungry\n 3.Thinking\n");
printf("\n Enter your choice\n");
printf("\t\t");
scanf("%d",&ch);
}
}
```

OUTPUT :

```
                Dining Philosopher Problem
                Initial State of Each Philosopher

        Philosopher No.         Think/Eat      Status          spoon
                6

1.Exit
2.Hungry
3.Thinking

Enter your choice
                2

        Enter which philosopher is hungry
                4

        State of Each philosopher


        Philosoper No.  Thinking        Hungry
                1               t               s       s

                2               t               s       s

                3               t               s       n

                4               e               s       n

                5               t               s       s
```

```
1.Exit
2.Hungry
3.Thinking

Enter your choice
                2

        Enter which philosopher is hungry
                3

        State of Each philosopher


        Philosoper No.  Thinking        Hungry
                1               t               s       s

                2               t               s       s

                3               h               w       n

                4               e               s       n

                5               t               s       s
```

RESULT :

EXP NO : 8 **DATE :**

IMPLEMENTATION OF BANKER'S ALGORITHM

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int i,j,z;
int res[10];
int resource,process,boolean=0;
int
allocation[10][10],sel[10],max[10][10],need[10][10],work[10];
int check=0;
printf("Welcome to Bankers Algorithms");
printf("\n Enter the no .of processes:");
scanf("%d",&process);
printf("\n Enter the number of Resources");
scanf("%d",&resource);
for(i=0;i<resource;i++)
{
printf("Enter the instances of Resource %d:",i+1);
scanf("%d",&res[i]);
}
for(i=0;i<process;i++)
{
printf("\n Enter allocated resources for process %d:",i+1);
for(j=0;j<resource;j++)
{
printf("\n Resource %d:",j+1);
scanf("%d",&allocation[i][j]);
}
```

```
}
for(i=0;i<process;i++)
sel[i]=-1;
for(i=0;i<process;i++)
{
printf("Enter maximum need of process: %d",i+1);
for(j=0;j<resource;j++)
{
printf("\n Resource %d:",j+1);
scanf("%d",&max[i][j]);
}
}
for(i=0;i<process;i++)
for(j=0;j<resource;j++)
need[i][j]=max[i][j]-allocation[i][j];
printf("\n The Need is \n");
for(i=0;i<process;i++)
{
for(j=0;j<resource;j++)
{
printf("\t%d",need[i][j]);
}
printf("\n");
}
printf("\n The Avalilable is ");
for(i=0;i<process;i++)
{
work[i]=0;
for(j=0;j<resource;j++)
work[i]=work[i]+allocation[j][i];
```

```
work[i]=res[i]-work[i];
printf("\t%d",work[i]);
}
for(i=0;i<process;i++)
{
for(j=0;j<resource;j++)
{
if(work[j]>=need[i][j]&&sel[i]==-1)
{
sel[i]=1;
work[j]=work[j]+allocation[i][j];
}
}}
for(i=0;i<process;i++)
{
if(sel[i]==1)
check=check+1;
else
{
check=-1;
}
}
if(check==process)
printf("\n System is in safe mode\n");
else
printf("\n System is in unsafe mode\n");
}
```

OUTPUT :

```
Welcome to Bankers Algorithms
 Enter the no .of processes:3

 Enter the number of Resources3
Enter the instances of Resource 1:3
Enter the instances of Resource 2:4
Enter the instances of Resource 3:2

 Enter allocated resources for process 1:
 Resource 1:1

 Resource 2:1

 Resource 3:1

 Enter allocated resources for process 2:
 Resource 1:1

 Resource 2:0

 Resource 3:1

 Enter allocated resources for process 3:
 Resource 1:1

 Resource 2:2

 Resource 3:0
Enter maximum need of process: 1
 Resource 1:1

 Resource 2:0

 Resource 3:0
Enter maximum need of process: 2
 Resource 1:1

 Resource 2:0

 Resource 3:1
Enter maximum need of process: 3
 Resource 1:0

 Resource 2:1

 Resource 3:0
```

```
The Need is
        0        -1       -1
        0        0        0
        -1       -1       0

The Avalilable is       0        1        0
System is in safe mode
```

RESULT :

EXP NO : 9A **DATE :**

FIRST IN FIRST OUT PAGE REPLACEMENT

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
 main()
{
int main_mem,cur=0,i=0,j,fault=0;
static int page[100],page_mem[100],flag,num;
for(i=0;i<100;i++)
      page_mem[i]=-2;
printf("\n\t\t\t\t paging->fifo\n");
printf("\n\n Enter the number of pages in main memory ");
scanf("%d",&main_mem);
printf("\n enter no of page references");
scanf("%d",&num);
for(i=0;i<num;i++)
{
      printf("\n Enter page reference:");
      scanf("%d",&page[i]);
}
printf("\n\t\t\t\t Fifo-> paging \n\n\n");
for(i=0;i<main_mem;i++)
      printf("\t page %d",i+1);
for(i=0;i<num;i++)
{
for(j=0;j<main_mem;j++)
if(page[i]==page_mem[j])
{
flag=1;
break;
}
```

```
if(!flag)
{
page_mem[cur]=page[i];
fault++;
}
printf("\n\n");
for(j=0;j<main_mem;j++)
      printf("\t%d",page_mem[j]);
if(!flag&&cur<main_mem-1)
cur++;
else if(!flag)
cur=0;
flag=0;
}
printf("\n\n-2 refers to empty blocks\n\n")
printf("\n\n No of page faults:%d\n",fault);
}
```

INPUT :

```
Enter the number of pages in main memory: 3
 enter no of page references: 8
 Enter page reference: 2
 Enter page reference: 0
 Enter page reference: 3
 Enter page reference: 0
 Enter page reference: 2
 Enter page reference: 3
 Enter page reference: 5
 Enter page reference: 9
```

OUTPUT :

```
                                paging->fifo

    Enter the number of pages in main memory 3
    enter no of page references8
    Enter page reference:2
    Enter page reference:0
    Enter page reference:3
    Enter page reference:0
    Enter page reference:2
    Enter page reference:3
    Enter page reference:5
    Enter page reference:9_
            page 1  page 2  page 3
          2       -2      -2
          2       0       -2
          2       0       3
          2       0       3
          2       0       3
          2       0       3
          5       0       3
          5       9       3
   -2 refers to empty blocks

    No of page faults:5
```

RESULT :

EXP NO : 9B **DATE :**

LEAST RECENTLY USED PAGE REPLACEMENT

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
#include<stdlib.h>
#define max 100
int frame[10],count[10],cstr[max],tot,nof,fault;
main()
{
 getdata();
push();
}
getdata()
{
      int pno,i=0;
      printf("\n\t\tL R U - Page Replacement Algorithm\n");
      printf("\nEnter No. of Pages in main memory:");
      scanf("%d",&nof);
      printf("\nEnter the no of page references:\n");
      scanf("%d",&pno);
      for(i=0;i<pno;i++)
      {printf("Enter page reference%d:",i);
      scanf("%d",&cstr[i]);
      }
      tot=i;
      for(i=0;i<nof;i++)
      printf("\tpage%d\t",i);
}
push()
{
      int x,i,j,k,flag=0,fault=0,nc=0,mark=0,maximum,maxpos=-1;
```

```
    for(i=0;i<nof;i++)
{
      frame[i]=-1;
         count[i]=mark--;
    }
    for(i=0;i<tot;i++)
    {
         flag=0;
         x=cstr[i];
         nc++;
         for(j=0; j<nof; j++)
           {
              for(k=0; k<nof;k++)
                   count[k]++;
              if(frame[j]==x)
              {
                   flag=1;
                   count[j]=1;
                   break;
              }
         }
         if(flag==0)
         {
              maximum = 0;
              for(k=0;k<nof;k++)
              {
                   if(count[k]>maximum && nc>nof)
                   {
                        maximum=count[k];
                        maxpos = k;
```

```
                }
            }
            if(nc>nof)
{
                frame[maxpos]=x;
                count[maxpos]=1;  }
             else
                    frame[nc-1]=x;
            fault++;
            dis();
        }
    }
    printf("\nTotal Page Faults :%d",fault);
}
dis()
{
    int i=0;
    printf("\n\n");
    while(i<nof)
    {
        printf("\t%d\t",frame[i]);
        i++;
    }
}
```

INPUT :

Enter the number of pages in main memory: 3

Enter the no of page references: 8

Enter page reference0: 0

Enter page reference1: 1

Enter page reference2: 2

Enter page reference3: 1

Enter page reference4: 2

Enter page reference5: 5

Enter page reference6: 0

Enter page reference7: 1

OUTPUT :

```
                L R U - Page Replacement Algorithm

Enter No. of Pages in main memory:3

Enter the no of page references:
8
Enter page reference0:0
Enter page reference1:1
Enter page reference2:2
Enter page reference3:1
Enter page reference4:2
Enter page reference5:5
Enter page reference6:0
Enter page reference7:1_

        page0           page1           page2

        0               -1              -1

        0               1               -1

        0               1               2

        5               1               2

        5               0               2

        5               0               1
Total Page Faults :6
```

RESULT :

EXP NO : 10A **DATE :**

FIRST FIT MEMORY ALLOCATION

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
#include<string.h>
main()
{
int n,j,i,size[10],sub[10],f[10],m,x,ch,t;
int cho;
printf("\t\t MEMORY MANAGEMENT \n");
printf("\t\t ================\n");
printf("\tEnter the total no of blocks: ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n Enter the size of blocks: ");
scanf("%d",&size[i]);
}
cho=0;
while(cho==0)
{
printf("\n Enter the size of the file: ");
scanf("%d",&m);
x=0;
for(i=1;i<=n;i++)
{
if(size[i]>=m)
{
printf("\n size can occupy %d",size[i]);
size[i]-=m;
x=i;
```

```
break;
}
}
if(x==0)
{
printf("\n\nBlock can't occupy\n\n");
}
printf("\n\nSNO\t\tAvailable block list\n") ;
for(i=1;i<=n;i++)
printf("\n\n%d\t\t\t%d",i,size[i]);
printf("\n\n Do u want to continue.....(0-->yes/1-->no): ");
scanf("%d",&cho);
}
}
```

INPUT :

Enter the total no of blocks: 4

Enter the size of blocks: 50

Enter the size of blocks: 20

Enter the size of blocks: 30

Enter the size of blocks: 40

Enter the size of the file: 25

OUTPUT :

```
                    MEMORY MANAGEMENT
                    =================
          Enter the total no of blocks: 4
 Enter the size of blocks: 50
 Enter the size of blocks: 20
 Enter the size of blocks: 30
 Enter the size of blocks: 40
 Enter the size of the file: 25
 size can occupy 50
SNO              Available block list

1                        25
2                        20
3                        30
4                        40
 Do u want to continue.....(0-->yes/1-->no): 0
 Enter the size of the file: 28
 size can occupy 30
SNO              Available block list

1                        25
2                        20
3                        2
4                        40
 Do u want to continue.....(0-->yes/1-->no): 0
 Enter the size of the file: 32
 size can occupy 40
SNO              Available block list

1                        25
2                        20
3                        2
4                        8
 Do u want to continue.....(0-->yes/1-->no): 1_
```

RESULT :

EXP NO : 10B **DATE :**

BEST FIT MEMORY ALLOCATION

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
main()
{
int n,j,i,size[10],sub[10],f[10],m,x,ch,t;
int cho;
printf("\t\t MEMORY MANAGEMENT \n");
printf("\t\t =================\n");
printf("\tENTER THE TOTAL NO OF blocks : ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\n Enter the size of blocks : ");
scanf("%d",&size[i]);
}
cho=0;
while(cho==0)
{
printf("\n Enter the size of the file : ");
scanf("%d",&m);
for(i=1;i<=n;i++)
{
sub[i]=size[i];
f[i]=i;
}
for(i=1;i<=n;i++)
{
for(j=i+1;j<=n;j++)
if(sub[i]>sub[j])
```

```
{
t=sub[i];
sub[i]=sub[j];
sub[j]=t;
t=f[i];f[i]=f[j];
f[j]=t;
}
}
for(i=1;i<=n;i++)
{
if(size[f[i]]>=m)
{
printf("size can occupy %d : ",size[f[i]]);
size[f[i]]-=m;
x=i;
break;
}
}
if(x==0)
{
printf("block can't occupy");
}
printf("\n\nSNO\t\t Available Block size\n") ;
for(i=1;i<=n;i++)
printf("\n%d\t\t%d",i,size[i]);
printf("\n\n Do u want to continue.....(0-->yes\t/1-->no)");
scanf("%d",&cho);
}
```

INPUT :

Enter the total no of blocks : 4

Enter the size of blocks : 50

Enter the size of blocks : 100

Enter the size of blocks : 200

Enter the size of blocks : 150

Enter the size of the file : 95

OUTPUT :

```
                    MEMORY MANAGEMENT
                    =================
          ENTER THE TOTAL NO OF blocks : 4

   Enter the size of blocks : 50

   Enter the size of blocks : 100

   Enter the size of blocks : 200

   Enter the size of blocks : 150

   Enter the size of the file : 95
  size can occupy 100 :
```

```
   Enter the size of the file : 95
  size can occupy 100 :

  SNO              Available Block size

  1               50
  2               5
  3               200
  4               150

   Do u want to continue.....(0-->yes     /1-->no)0

   Enter the size of the file : 48
  size can occupy 50 :

  SNO              Available Block size

  1               2
  2               5
  3               200
  4               150

   Do u want to continue.....(0-->yes     /1-->no)1_
```

RESULT :

EXP NO : 10C **DATE :**

WORST FIT MEMORY ALLOCATION

AIM :

ALGORITHM :

PROGRAM :

```
#include<stdio.h>
int p[10]={0},m=0,x=0,b[10]={0},a[10]={0};
main()
{
int j=0,n=0,pra[10]={0},pro[10]={0},z[10],ch,flag,flag2,sum=0,sum2=0;
int c=0,i=0,k=0;
printf("\n\t\tMEMORY MANAGEMENT POLICIES\n");
printf("\n enter the no of process:\t");
scanf("%d",&n);
printf("\n enter the no of partition:\t");
scanf("%d",&m);
printf("\nprocess information\n");
for(i=0;i<n;i++)
{
printf("\n enter the memory required for process P%d:",i+1);
scanf("\t%d",&a[i]);
pro[i]=a[i];
}
printf("\n memory partition information\n");
for(j=0;j<m;j++)
{
printf("\n enter the block size of block B%d:",j+1);
scanf("\t%d",&p[j]);
pra[j]=p[j];
}
arrange();
printf("\n process partition\n\n");
for(i=0;i<n;i++)
```

```
{
for(j=0;j<m;j++)
{
if(a[i]<p[j])
{
printf("%d\t%d\n",a[i],p[j]);
p[j]=0;
flag=i;
break;
}
}
if(flag!=i)
printf("%d\t%s\n",a[i],"waiting");
arrange();
}
}
arrange()
{
int i,j,t;
for(i=0;i<m-1;i++)
for(j=0;j<m-i-1;j++)
{
if(p[j]<p[j+1])
{
t=p[j+1];
p[j+1]=p[j];
p[j]=t;
}
}
}
```

INPUT :

MEMORY MANAGEMENT POLICIES

enter the no of process: 5

enter the no of partition: 5

process information

enter the memory required for process P1:212

enter the memory required for process P2:417

enter the memory required for process P3:112

enter the memory required for process P4:321

enter the memory required for process P5:460

memory partition information

enter the block size of block B1:400

enter the block size of block B2:300

enter the block size of block B3:500

enter the block size of block B4:200

enter the block size of block B5:600

OUTPUT :

```
                    MEMORY MANAGEMENT POLICIES

    enter the no of process:       5

    enter the no of partition:     5

   process information

    enter the memory required for process P1:212

    enter the memory required for process P2:417

    enter the memory required for process P3:112

    enter the memory required for process P4:321

    enter the memory required for process P5:460_
```

```
    memory partition information

    enter the block size of block B1:
                                     400

    enter the block size of block B2:300

    enter the block size of block B3:500

    enter the block size of block B4:200

    enter the block size of block B5:600

    process partition

   212     600
   417     500
   112     400
   321     waiting
   460     waiting
```

RESULT :