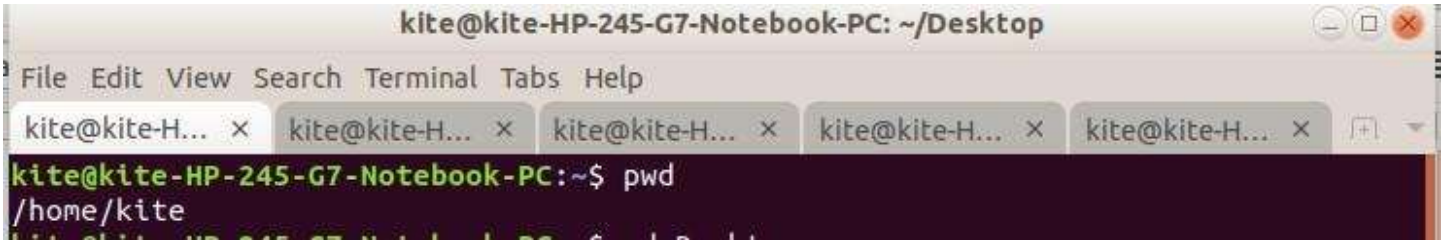
OPERATING SYSTEM RECORD

Submitted by,
Joyal Raphel
CSE B
Roll:04

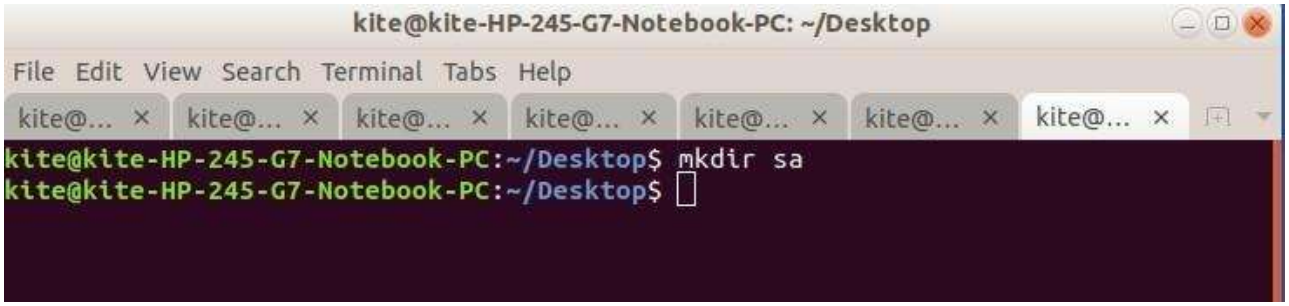
Sl.no	command	description	Synatx
1	pwd	**pwd** stands for Print Working Directory. It prints the path of the working directory, starting from the root	pwd
2	mkdir	create directories	mkdir directoryname
3	rmdir	Remove the directory	rmdir directoryname
4	cd	Change directory	cd directoryname
5	cd ~	To navigate to Home directory	cd ~
6	cd ..	To navigate to parent directory	cd ..
7	cd -	To navigate to previous directory	cd -
8	cat	To copy the contents from one file to another file	cp sourcefile destinationfile
9	cp	To copy the contents from one file to another file	cp sourcefile destinationfile
10	clear	To clear the terminal	clear
11	cmp	To compare two given files	cmp firstfile secondfile
12	ls	To list all the files in the directory	ls
13	ls -al	To list all files with all details of each file	ls -al
14	ls -r	To list the files in reverse order	ls -r
15	ls -a	To list all files including hidden files	ls -a
16	ls -R	To produce a recursive listing	ls –R
17	ls -l	To give a detailed or long listing of all files	ls –l
18	mv	Move or rename files	mv firstfile secondfile
19	echo	Display string passed as argument	echo ‘string'
20	uname	To print the name of operating system being used	Uname
21	who	To check logged in users	Who
22	cal	To display calendar	Cal

23	history	To display previous commands used	History
24	date	To display current date and time	Date
25	rm	To remove a given file	rm filename
26	sort	To sort the contents of a file line by line	sort filename
27	ping	To check whether a host is reachable	ping google.com
28	reboot	To reboot the system	Reboot
29	man	To display the manual of a command	man command
30	last	To display list of last logged in users	Last

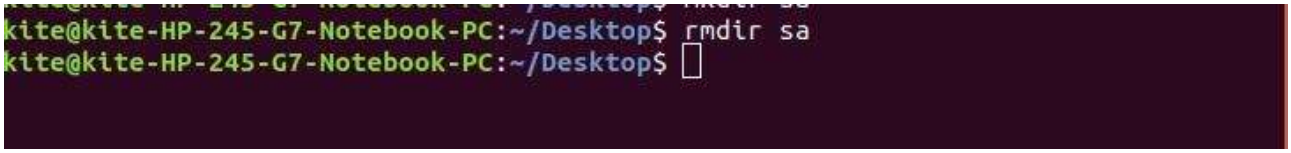
pwd



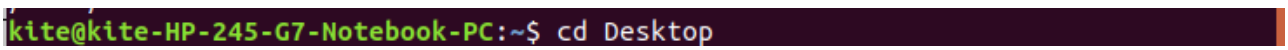
mkdir



rmdir



cd



cd ~ , cd - , cd ..

```
kite@kite-HP-245-G7-Notebook-PC:~$ cd Desktop
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cd ~
kite@kite-HP-245-G7-Notebook-PC:~$ cd Desktop
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cd ..
kite@kite-HP-245-G7-Notebook-PC:~$ cd Desktop
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cd -
/home/kite
```

cat

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat su.txt sa.txt
12sa
csb
12sa
csb
```

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat>f1.txt
```

cp

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cp su.txt sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$
```

clear

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ clear
```

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls
 1.png
 2.jpg
 3A
'3A students work'
'Day 1'
'Day 2'
'Day 3'
'Digital Portfolio'
 f1.txt
 g.pdf
'java tutorial.pdf'
 kvh
```

ls -al

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -al
total 680
drwxr-xr-x 12 kite kite  12288 May  1 19:12  .
drwxr-xr-x 41 kite kite   4096 Apr 30 13:57  ..
-rw-r--r--  1 kite kite 105263 Mar 21  2020  1.png
-rw-r--r--  1 kite kite   8906 Mar 21  2020  2.jpg
drwxr-xr-x  2 kite kite  20480 Aug 31  2020  3A
drwxr-xr-x 38 kite kite   4096 Jul  4  2020 '3A students work'
drwxr-xr-x  6 kite kite   4096 Mar 19  2020 'Day 1'
drwxr-xr-x  8 kite kite   4096 Mar 19  2020 'Day 2'
drwxr-xr-x  6 kite kite   4096 Mar 19  2020 'Day 3'
drwxr-xr-x  2 kite kite   4096 Sep 30  2020 'Digital Portfolio'
-rw-r--r--  1 kite kite      0 May  1 19:12  f1.txt
```

ls -r

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -r
'[illegible] 3F'
 Zack.Snyders.Justice.League.2021.720p.HMAX.WEB-DL.DDP5.1.Atmos.H.264-MZABI-Engl
ish.srt
'Untitled Document'
'Untitled 2.odt'
'Untitled 1.odt'
 su.txt
 so
 sa.txt
```

ls -a

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -a
 .
 ..
 1.png
 2.jpg
 3A
'3A students work'
'Day 1'
'Day 2'
'Day 3'
'Digital Portfolio'
```

ls -R

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -R
.:
 1.png
 2.jpg
 3A
'3A students work'
'Day 1'
'Day 2'
'Day 3'
'Digital Portfolio'
 f1.txt
 g.pdf
'java tutorial.pdf'
```

ls -l

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l
total 660
-rw-r--r--  1 kite kite 105263 Mar 21  2020  1.png
-rw-r--r--  1 kite kite   8906 Mar 21  2020  2.jpg
drwxr-xr-x  2 kite kite  20480 Aug 31  2020  3A
drwxr-xr-x 38 kite kite   4096 Jul  4  2020 '3A students work'
drwxr-xr-x  6 kite kite   4096 Mar 19  2020 'Day 1'
drwxr-xr-x  8 kite kite   4096 Mar 19  2020 'Day 2'
drwxr-xr-x  6 kite kite   4096 Mar 19  2020 'Day 3'
drwxr-xr-x  2 kite kite   4096 Sep 30  2020 'Digital Portfolio'
-rw-r--r--  1 kite kite      0 May  1 19:12  f1.txt
-rw-r--r--  1 kite kite  28798 Jul 13  2020  g.pdf
```

mv

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ mv su.txt sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$
```

echo

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ echo '123'
123
```

uname

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ uname
Linux
```

who

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ who
kite     tty7         2021-04-30 13:57 (:0)
```

cal

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cal
      May 2021
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

history

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ history
    1  shutdown -h 00
    2  gedit exp1.c
    3  gcc exp1.c
    4  gcc exp1.c -o main -lm
    5  ./a.out
    6  gcc exp1.c -o exp1 -lm
    7  ./a.out
    8  gcc
    9  gedit Test.c
   10  gcc Test.c
   11  gedit Test.c &
   12  gcc Test.c
   13  ./a.out
   14  gedit Test.c &
   15  gcc Test.c
   16  ./a.out
   17  gcc Test.c
   18  ./a.out
   19  gcc Test.c
   20  ./a.out
```

date

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ date
Sat May  1 20:33:43 IST 2021
```

rm

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ rm f1.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$
```

sort

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ sort sa.txt
12sa
csb
```

ping

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ping google.com
PING google.com(maa03s38-in-x0e.1e100.net (2404:6800:4007:823::200e)) 56 data by
tes
64 bytes from maa03s38-in-x0e.1e100.net (2404:6800:4007:823::200e): icmp_seq=1 t
tl=114 time=138 ms
64 bytes from maa03s38-in-x0e.1e100.net (2404:6800:4007:823::200e): icmp_seq=2 t
tl=114 time=133 ms
64 bytes from maa03s38-in-x0e.1e100.net (2404:6800:4007:823::200e): icmp_seq=3 t
tl=114 time=94.7 ms
64 bytes from maa03s38-in-x0e.1e100.net (2404:6800:4007:823::200e): icmp_seq=4 t
tl=114 time=49.7 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 49.768/104.148/138.458/35.664 ms
```

reboot

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ reboot
```

man

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
 Manual page ls(1) line 1 (press h for help or q to quit)
```

last

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ last

wtmp begins Sat May  1 20:05:35 2021
```

Sl.no	command	description	Synatx
31	whereis	It is used to locate the binary source and manual page files for any command	whereis command
32	script	It is used to make transcript or record all the terminal activities	script filename
33	bc	It is used to activate the command line calculator	Bc
34	apt-get	it is used to install new software packages, remove or upgrade existing packages	sudo apt-get install packagename
35	chmod	It is used to change the access mode of a file	chmod a+x file.txt
36	ls > file.txt	This command isexecuted and the results are written in a file name file.txt	echo hello > file.txt cat file.txt
37	ls >> file.txt	The new results are added to the end of an existing file.	echo hello >> file.txt catfile.txt
38	wc < file.txt	This command is used to display no.of words, lines and characters on the terminal	wc < file.txt
39	grep	This command is used to search for a string of characters in a specified file	grep string file.txt
40	less	Displays the contents of a file or a command output,one page at a time	less filename
41	head	Displays the first n lines of the specified text file	ls -l | head -n ls as an example
42	tail	It returns the lines from bottom to up	ls -l | tail -n ls as an example
43	uniq	Used to remove all the repeated lines in a file	uniq file.txt

44	cut	Cutting out the sections from each line of files and writing the result to standard output	cut -d ‘c ‘ -f file.txt c is delimiter

OUTPUT

whereis

```
kite@kite-HP-245-G7-Notebook-PC:~$ whereis bash
bash: /bin/bash /etc/bash.bashrc /usr/share/man/man1/bash.1.gz
kite@kite-HP-245-G7-Notebook-PC:~$
```

ls >> file.txt

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls >> sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat sa.txt
12sa
csb
1.png
2.jpg
3A
3A students work
Day 1
Day 2
Day 3
Digital Portfolio
file.txt
g.pdf
java tutorial.pdf
```

ls > file.txt

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls > sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat sa.txt
1.png
2.jpg
3A
3A students work
Day 1
Day 2
Day 3
Digital Portfolio
file.txt
g.pdf
java tutorial.pdf
kvh
```

chmod

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l sa.txt
-rw-r--r-- 1 kite kite 385 May  8 21:49 sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ chmod o+x sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l sa.txt
-rw-r--r-x 1 kite kite 385 May  8 21:49 sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ chmod g+x sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l sa.txt
-rw-r-xr-x 1 kite kite 385 May  8 21:49 sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ chmod g+w sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l sa.txt
-rw-rwxr-x 1 kite kite 385 May  8 21:49 sa.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$
```

bc

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
 Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
1+7
8
quit
```

script

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat typescript
Script started on 2021-05-08 21:58:27+0530
```

wc and grep

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat > w.txt
hello
sa
d
^C
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ wc -l w.txt
3 w.txt
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ | grep hello w.txt
bash: syntax error near unexpected token `|'
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ grep hello w.txt
hello
```

head,tail,uniq,cut

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l | head -3
total 1800
-rw-r--r--  1 kite kite 105263 Mar 21  2020 1.png
-rw-r--r--  1 kite kite   8906 Mar 21  2020 2.jpg
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ls -l | tail -3
-rw-r--r--  1 kite kite     11 May  8 22:05 w.txt
-rw-r--r--  1 kite kite 187064 Mar 18 14:52 Zack.Snyders.Justice.League.2021.720
p.HMAX.WEB-DL.DDP5.1.Atmos.H.264-MZABI-English.srt
drwxr-xr-x  2 kite kite  24576 Jun 27  2020 comics 3F
```

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ uniq w.txt
hello
sa
d
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cat > ex.txt
s,jk,hgj
jkhg,kjuh
^C
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ cut -d ',' -f1 ex.txt
s
jkhg
```

less

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ less w.txt
```

```
hello
sa
d
w.txt (END)
```

SHELL SCRIPT

1.Read name,age,city and print it

```
Open
echo "name?"
read a;
echo "age?"
read b;
echo "city?"
read c;
echo "$a $b $c"
```

```
kite@kite-HP-245-G7-Notebook-PC:~$ gedit name.sh
^C
kite@kite-HP-245-G7-Notebook-PC:~$ chmod 777 name.sh
kite@kite-HP-245-G7-Notebook-PC:~$ ./name.sh
name?
saheen
age?
20
city?
malappuram
saheen 20 malappuram
kite@kite-HP-245-G7-Notebook-PC:~$
```

2.Find the difference between two numbers

```
read a
read b
((c=a-b))
echo $c
```

```
kite@kite-HP-245-G7-Notebook-PC:~$ gedit diff.sh
kite@kite-HP-245-G7-Notebook-PC:~$ chmod 777 diff.sh
kite@kite-HP-245-G7-Notebook-PC:~$ ./diff.sh
20
5
15
```

3.Find the smallest of two numbers

```
read a
read b
if((a<b))
then echo $a
else echo $b
fi
```

```
kite@kite-HP-245-G7-Notebook-PC:~$ chmod 777 loop.sh
kite@kite-HP-245-G7-Notebook-PC:~$ ./loop.sh
9
4
4
```

4.Write shell script to show various system configuration like

.your current shell

.your home directory

.your current working directory

```
echo "my current shell:$SHELL"
echo "my home directory:$HOME"
echo "my current working directory:$PWD"
```

```
kite@kite-HP-245-G7-Notebook-PC:~$ gedit dir.sh
kite@kite-HP-245-G7-Notebook-PC:~$ chmod 777 dir.sh
kite@kite-HP-245-G7-Notebook-PC:~$ ./dir.sh
my current shell:/bin/bash
my home directory:/home/kite
my current working directory:/home/kite
```

SHELL PROGRAMING

1. Factorial using function

AIM:

Write a shell script program to implement factorial using function.

ALGORITHM:

1.start
2.define function factorial
  2.1 if (input number is greater than 1)
    2.1.1 store previous number in i
    2.1.2 call the function factorial with j
    2.1.3 multiply i with j and store it in k
    2.1.4 print k
  2.2 else
    2.2.1 print 1
  2.3 endif
3.print”enter a number”
4.read the number
5.call the function factorial with the number
6.stop

CODE:

```
factorial()
{
 if [ "$1" -gt "1" ]; then
   i=`expr $1 - 1`
   j=`factorial $i`
   k=`expr $1 \* $j`
   echo $k
 else
   echo 1
 fi
}
 echo "Enter a number:"
 read a
 echo "factorial of number is:"
 factorial $a
```

OUTPUT:

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ./factorial.sh
Enter a number:
5
factorial of number is:
120
```

2.Greatest of three numbers using function

AIM:

Write a shell script to find greatest of three numbers using function.

ALGORITHM:

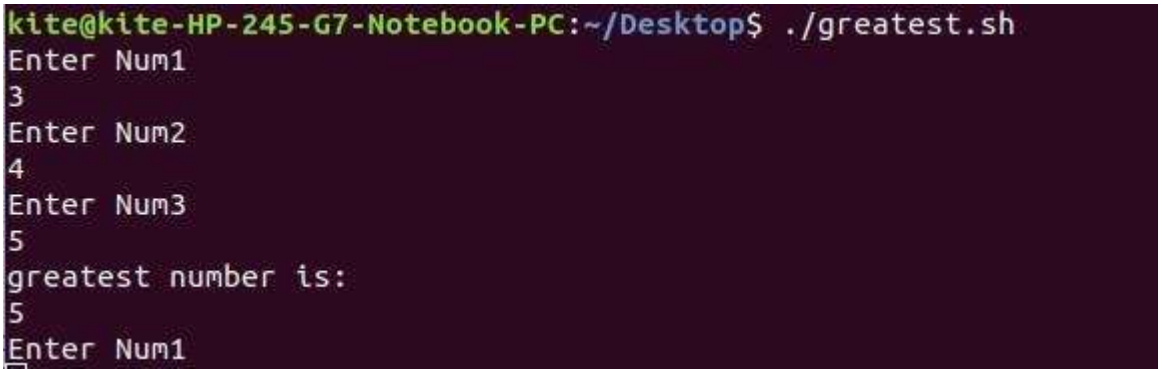
1.start
2.define function greater
  2.1 if (num 1 > num 2) && (num 1 > num 3)
      2.1.1 print num 1
  2.2 elif (num 2 > num 1) && (num 2 > num 3)
      2.2.1 print num 2
  2.3 else
      2.3.1 print num 3
  2.4 endif
3.print “enter the numbers”
4.read the numbers
5.call function using the three numbers
6.stop

CODE:

```
greater()
{
 if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
    echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
    echo $num2
else
    echo $num3
fi
}

while :
do
  echo "Enter Num1"
   read num1
  echo "Enter Num2"
   read num2
  echo "Enter Num3"
   read num3
  echo "greatest number is:"
greater $num1 $num2 $num3
done
```

OUTPUT:



3.Average of n numbers

AIM:

Write a shell program to find average of n numbers.

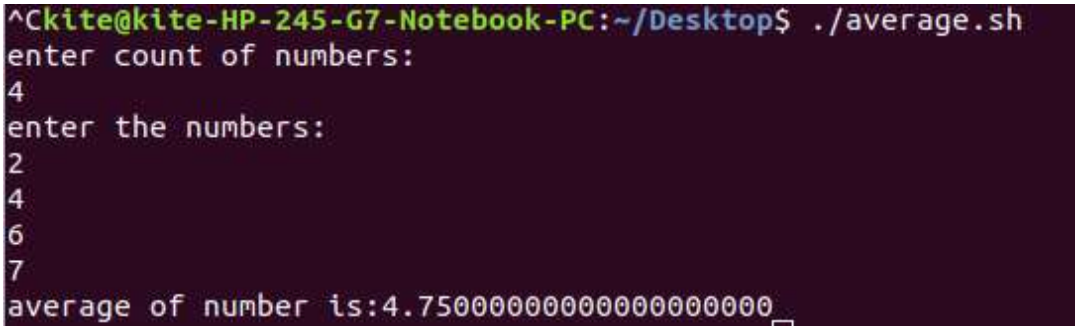
ALGORITHM:

1.start
2.print “enter the count of numbers”
3.read the count
4.for(i=1;i<=count;i++)
   4.1 read the numbers
   4.2 sum=sum+input numbers (initial value sum=0)
5.average=sum/count
6.print average
7.stop

CODE:

```
echo "enter count of numbers:"
read a
sum=0
echo "enter the numbers:"
for((i=1;i<=a;i++))
do
read n
sum=$((sum+n))
done
avg=$(echo $sum/$a | bc -l )
echo "average of number is:$avg"
```

OUTPUT:



4.Sum of n numbers

AIM:

Write a shell program to find sum of n numbers.

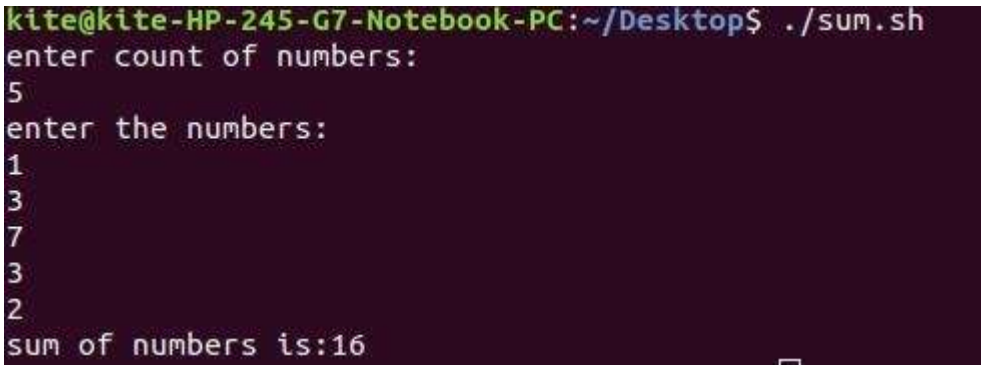
ALGORITHM:

1.start
2.print “enter the count of numbers”
3.read the count
4.for(i=1;i<=count;i++)
  4.1 read the numbers
  4.2 sum=sum+input numbers (initial value sum=0)
5.endfor
6.print sum

CODE:

```
echo "enter count of numbers:"
read a
sum=0
echo "enter the numbers:"
for((i=1;i<=a;i++))
do
read n
sum=$((sum+n))
done
echo "sum of numbers is:$sum"
```

OUTPUT:



5.GCD of two numbers

AIM:

Write a shell program to find gcd of 2 numbers.

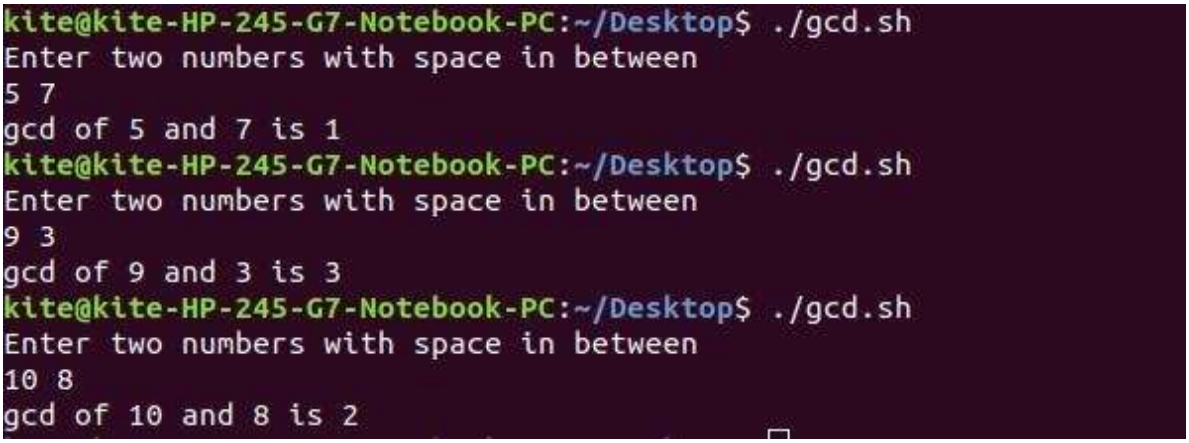
ALGORITHM

1.start
2.print “enter the two numbers”
3.read the two numbers
4.m=first number
5.if(second number < m)
  5.1 m=second number
  6.endif
7.while(m != 0)
  7.1 x=first number % m
  7.2 y=second number % m
  7.3 if (if [$x -eq 0 -a $y -eq 0])
  7.4 gcd is a(first number)
  7.5 endwhile
8.m=m-1
9.stop

CODE:

```
echo "Enter two numbers with space in between"
read a b
m=$a
if [ $b -lt $m ]
then
m=$b
fi
while [ $m -ne 0 ]
do
x=`expr $a % $m`
y=`expr $b % $m`
if [ $x -eq 0 -a $y -eq 0 ]
then
echo gcd of $a and $b is $m
break
fi
m=`expr $m - 1`
done
```

OUTPUT:



6.System configuration

AIM:

Write shell script to show various system configuration like currently logged user and his login name

- Your current shell
- Your home directory
- Your operating system type
- Your current path setting
- Your current working directory
- Show Currently logged number of users

ALGORITHM:

1.start
2.print
"my current shell:$SHELL"
"my home directory:$HOME"
"my current working directory:$PWD"
"os type:$(uname)"
"users:$(who)"
"current path setting:$PATH"
3.stop

CODE:

```
echo "my current shell:$SHELL"
echo "my home directory:$HOME"
echo "my current working directory:$PWD"
echo  "os type:$(uname)"
echo "users:$(who)"
echo "current path setting:$PATH"
```

OUTPUT:

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ./man.sh
my current shell:/bin/bash
my home directory:/home/kite
my current working directory:/home/kite/Desktop
os type:Linux
users:kite     tty7         2021-06-20 19:41 (:0)
current path setting:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n:/usr/games:/usr/local/games:/snap/bin
```

7.calculator

AIM:
Write a shell script to implement a menu driven calculator with following functions
1. Addition 2. Subtraction 3. Multiplication 4. Division 5. Modulus
ALGORITHM:
1. start
2. print “enter the first number”
3.read first number ie a
4.print “enter the second number”
5.read second number ie b
6. choose the operation to perform
1. addition 2.subtraction 3.multiplication 4.division 5.modulus
7.read the choice
8. case choise in
8.1 1)sum= a+b
8.2 2)sub=a-b
8.3 3)mul=a*b
8.4 4)div=a/b
8.5 5)mod=a%b
8.6 *)print “invalid choice”
9.stop

CODE:
```
echo -n "enter the first number:"
read a
echo -n "enter the second number:"
read b
i="y"
while [ $i = "y" ]
do
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "5.Modulus"
echo "Enter your choice"
read op
```

```
case $op in
1)sum=`expr $a + $b`
echo "Sum ="$sum;;
2)sub=`expr $a - $b`
echo "Sub = "$sub;;
3)mul=`expr $a \* $b`
echo "Mul = "$mul;;
4)div=`expr $a / $b`
echo "Div = "$div;;
5)mod=`expr $a % $b`
echo "mod = "$mod;;
*)echo "Invalid choice";;
esac
echo "Do u want to continue ?"
read i
if [ $i != "y" ]
then
exit
fi
done
```

OUTPUT:

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ./calc.sh
enter the first number:4
enter the second number:5
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
Enter your choice
1
Sum =9
Do u want to continue ?
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
5.Modulus
Enter your choice
5
mod = 4
Do u want to continue ?
```

1. Write a C program to create a child process that lists the files and directories and the
parent process waits till the child completes. Also print the PID’s of parent and child
process.

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>
int main()
{
pid_t p1;
p1 = fork();
if(p1<0)
{
printf("failed");
return 1;
}
else if(p1==0)
{
printf("Files list: \n");
execlp("/bin/ls","ls",NULL);
printf("\n");
}
else
{
wait(NULL);
printf("PID of Child = %d\n",getpid());
printf("PID of Parent = %d \n",getppid());
}
return 0;
}
```

OUTPUT:

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ gcc sys1.c
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ./a.out
Files list:
 1.png                          fork.c
 2.jpg                          gcd.sh
 3A                             greatest.sh
'3A students work'              kvh
 a.out                          less.sh
 average.sh                     man.sh
 bcd.odt                        mod.sh
 calc.sh                        new
 calculator.sh                  pos.sh
'csb 42 2b.odt'                 'roll 42 6a,6b.odt'
'csb 42 2d.odt'                 s
'csb 42 3b.odt'                 saheen
'csb 42 4.odt'                  'saheen csb roll 42.odt'
'csb 42 5a.odt'                 so
'csb 42  full addr nand.odt'    sum.sh
'csb 42 half addr nand.odt'     sys1.c
 c.sh                           sys2.c
'Day 1'                         sys3.c
'Day 2'                         systemcall1.c
'Day 3'                         systemcall2.c
'Digital Portfolio'             'Untitled 1.odt'
'exp 2 saheen usman 42.pdf'     working.sh
 fact                           'ഫസ്ട് 3F'
 factorial.sh
PID of Child = 2832
PID of Parent = 2787
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$
```

2. Write a C program to create 5 child processes and print the PID of each child and its
parent process.


```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
for (int i = 0; i < 5; i++)
{
if (fork() == 0)
{
printf("CHILD PID %d from PARENT PID %d\n", getpid(), getppid());
exit(0);
}
}
for (int i = 0; i < 5; i++)
{
wait(NULL);
}
return 0;
}
```

OUTPUT:

```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ gcc sys2.c
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ./a.out
CHILD PID 3001 from PARENT PID 3000
CHILD PID 3002 from PARENT PID 3000
CHILD PID 3004 from PARENT PID 3000
CHILD PID 3003 from PARENT PID 3000
CHILD PID 3005 from PARENT PID 3000
```

3. Write a C program using system calls stat, opendir, closedir to display the files and
their sizes.


```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <stdlib.h>
int main(int argc,char **argv)
{
struct stat buf;
int exists;
DIR *d;
struct dirent *de;
d=opendir(".");
if(d==NULL)
{
fprintf(stderr,"couldn't open \".\"\n");
exit(1);
}
for(de=readdir(d); de!=NULL;de=readdir(d))
{
exists=stat(de->d_name,&buf);
if (exists<0){
fprintf(stderr,"%s not found\n",de->d_name);
}
else
{
printf("%s %ld\n",de->d_name,buf.st_size);
}
}
closedir(d);
return 0;
}
```

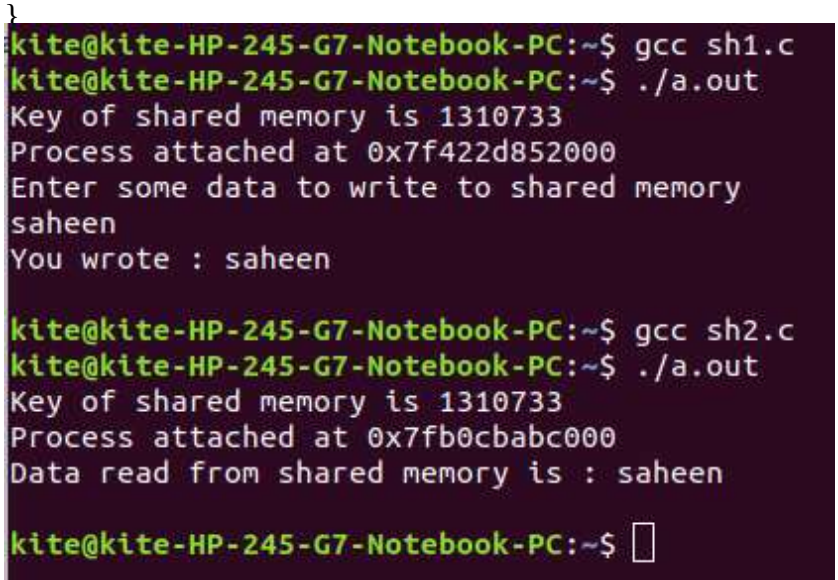
```
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ gcc sys3
kite@kite-HP-245-G7-Notebook-PC:~/Desktop$ ./a.out
. 12288
.. 12288
saheen 4096
csb 42  full addr nand.odt 9824
new 4096
gcd.sh 238
csb 42 4.odt 9988
csb 42 5a.odt 10640
c.sh 558
systemcall1.c 349
working.sh 187
mod.sh 104
Digital Portfolio 4096
saheen csb roll 42.odt 10934
average.sh 183
2.jpg 8906
sys2.c 390
[illegible] 3F 24576
Day 1 4096
```

sh1.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);
printf("Enter some data to write to shared memory\n");
read(0,buff,100);
strcpy(shared_memory,buff);
printf("You wrote : %s\n",(char *)shared_memory);
}
```

sh2.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);
printf("Data read from shared memory is : %s\n",(char *)shared_memory);
}
```



FCFS

code:

```
#include<stdio.h>
#include<string.h>
struct process
{
char pname[20];
int at,bt,wt,tt,status;
}p[20],t;
struct done
{
char name[20];
int st,ct;
} d[20];
void main()
{
int n,i,j,idle,k, num;
float sumwt=0.0, sumtt=0.0,st,w;
printf("\nEnter the number of processes : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the process name : ");
__fpurge(stdin);
gets(p[i].pname);
printf("\nEnter the arrival time : ");
scanf("%d",&p[i].at);
printf("\nEnter the burst time : ");
scanf("%d",&p[i].bt);
p[i].status=0;
}
for(i=0;i<n;i++)
{
for(j=0; j<n-i-1; j++)
{
if(p[j].at>p[j+1].at)
{
t=p[j];
p[j]=p[j+1];
p[j+1]=t;
}
}
}

idle=0;
for(i=0,k=0, num=0;k<n;)
{
if(p[k].at<=i&&p[k].status==0)
{
```

```
if(idle==1)
{
d[num].ct=i;
num++;
}
strcpy(d[num].name,p[k].pname);
d[num].st=i;
//printf("%d %d\n",d[num].st,i);
d[num].ct=i+p[k].bt;
p[k].tt=d[num].ct-p[k].at;
p[k].wt=p[k].tt-p[k].bt;
i=d[num].ct;
p[k].status=1;
k++;
num++;
idle=0;
}
else if(idle==0)
{
strcpy(d[num].name, "idle");
d[num].st=i;
i++;
idle=1;
}
else
{
i++;
}
}
printf (" |pname | arrival time\t | burst time\t | status\t | wait\t turn");
for (i =0;i<n; i++)
{
printf ("\n |%s \t |%d \t \t |%d \t \t |%d \t \t |%d \t | %d \n", p[i].pname ,p[i].at , p[i].bt ,
p[i].status,p[i].wt,p[i].tt);
}
//gant chart
printf("Gant chart\n");
printf("\
n----------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<num;++i){
printf("|%d |\t %s \t ",d[i].st,d[i].name);
}
printf("|%d|",d[num-1].ct);
printf("\
n----------------------------------------------------------------------------------------------------------------
----------------------\n");

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

```
sumwt=sumwt+p[i].wt;
sumtt=sumtt+p[i].tt;
}
w=(float)(sumwt/n);
st=(float)(sumtt/n);

printf("\nAverage waiting time=%f",w);
printf("\nAverage turnaround time=%f\n", st);
}
```


output:

```
Enter the number of processes : 4
Enter the process name : i1
Enter the arrival time : 0
Enter the burst time : 10
Enter the process name : p2
Enter the arrival time : 1
Enter the burst time : 6
Enter the process name : p3
Enter the arrival time : 3
Enter the burst time : 2
Enter the process name : p4
Enter the arrival time : 5
Enter the burst time : 4

|pname | arrival time | burst time | status | wait turn
|i1 |0 |10 |1 |0 | 10
|p2 |1 |6 |1 |9 | 15
|p3 |3 |2 |1 |13 | 15
|p4 |5 |4 |1 |13 | 17
Gant chart
-------------------------------------------------------------------------------------------------------------------------------
--------------------
|0 | i1 |10 | p2 |16 | p3 |18 | p4 |22|
-------------------------------------------------------------------------------------------------------------------------------
--------------------
Average waiting time=8.750000
Average turnaround time=14.250000
```

2: SJF

Code:

```
#include<stdio.h>
#include<string.h>
struct process
{
char pname[20];
int at,bt,wt,tt,status;
}p[20],t;
struct done
{
char name[20];
int st,ct;
} d[20];
void main()
{
int n,i,j,idle,k, num;
float sumwt=0.0, sumtt=0.0,st,w;
printf("\nEnter the number of processes : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the process name : ");
__fpurge(stdin);
gets(p[i].pname);
printf("\nEnter the arrival time : ");
scanf("%d",&p[i].at);
printf("\nEnter the burst time : ");
scanf("%d",&p[i].bt);
p[i].status=0;
//printf("checked");
}
//printf("hello");
//sorting in order of arrival time
for(i=0;i<n;i++)
{
for(j=0; j<n-i-1; j++)
{
if(p[j].at>p[j+1].at)
{
t=p[j];
p[j]=p[j+1];
p[j+1]=t;
}
}
}
//printf("hello");
//execution process
idle=0;
```

```
for(i=0,k=0, num=0;k<n;)
{
if(p[k].at<=i&&p[k].status==0)
{
if(idle==1)
{
d[num].ct=i;
num++;
}
int r=k+1;
while(r<n){
if(p[r].at<=i && p[r].status==0){
if(p[r].bt <p[k].bt){
t=p[r];
p[r]=p[k];
p[k]=t;
}
}
else{
break;
}
r++;
}
strcpy(d[num].name,p[k].pname);
d[num].st=i;
d[num].ct=i+p[k].bt;
p[k].wt=d[num].st-p[k].at;
p[k].tt=p[k].wt+p[k].bt;
i=d[num].ct;
p[k].status=1;
k++;
num++;
idle=0;
}
else if(idle==0)
{
strcpy(d[num].name, "idle");
d[num].st=i;
i++;
idle=1;
}
else
{
i++;
}
}
printf (" |pname | arrival time\t | burst time\t | status\t | wait\t turn");
for (i =0;i<n; i++)
{
printf ("\n |%s \t |%d \t \t |%d \t \t |%d \t \t |%d \t | %d \n", p[i].pname ,p[i].at , p[i].bt ,
```

```
p[i].status,p[i].wt,p[i].tt);

}
//gant chart
printf("Gant chart\n");
printf("\
n----------------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<num;++i){
printf("|%d |\t %s \t ",d[i].st,d[i].name);
}
printf("|%d|",d[num-1].ct);
printf("\
n----------------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<n; i++)
{
sumwt=sumwt+p[i].wt;
sumtt=sumtt+p[i].tt;
}
w=(float)(sumwt/n);
st=(float)(sumtt/n);

printf("\nAverage waiting time=%f",w);
printf("\nAverage turnaround time=%f\n", st);
}
```


Output:

```
Enter the number of processes : 4
Enter the process name : p1
Enter the arrival time : 0
Enter the burst time : 8
Enter the process name : p2
Enter the arrival time : 2
Enter the burst time : 4
Enter the process name : p3
Enter the arrival time : 4
Enter the burst time : 9
Enter the process name : p4
Enter the arrival time : 5

Enter the burst time : 5
|pname | arrival time | burst time | status | wait turn
|p1 |0 |8 |1 |0 | 8
|p2 |2 |4 |1 |6 | 10
|p4 |5 |5 |1 |7 | 12
|p3 |4 |9 |1 |13 | 22
Gant chart
----------------------------------------------------------------------------------------------------------------------
--------------------
|0 | p1 |8 | p2 |12 | p4 |17 | p3 |26|
```


Average waiting time=6.500000
Average turnaround time=13.000000

3:priority scheduling

Code:

```
#include<stdio.h>
#include<string.h>
struct process
{
char pname[20];
int at,bt,wt,tt,status,pr;
}p[20],t;
struct done
{
char name[20];
int st,ct;
} d[20];
void main()
{
int n,i,j,idle,k, num;
float sumwt=0.0, sumtt=0.0,st,w;
printf("\nEnter the number of processes : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the process name : ");
__fpurge(stdin);
gets(p[i].pname);
printf("\nEnter the arrival time : ");
scanf("%d",&p[i].at);
printf("\nEnter the burst time : ");
scanf("%d",&p[i].bt);
printf("\nEnter the priority : ");
scanf("%d",&p[i].pr);
p[i].status=0;
//printf("checked");
}
//printf("hello");
//sorting in order of arrival time
for(i=0;i<n;i++)
{
for(j=0; j<n-i-1; j++)
{
if(p[j].at>p[j+1].at)
{
t=p[j];
p[j]=p[j+1];
p[j+1]=t;
}
}
}
//printf("hello");
```

```
//execution process
idle=0;
for(i=0,k=0, num=0;k<n;)
{
if(p[k].at<=i&&p[k].status==0)
{
if(idle==1)
{
d[num].ct=i;
num++;
}
int r=k+1;
while(r<n){
if(p[r].at<=i && p[r].status==0){
if(p[r].pr <p[k].pr){
t=p[r];
p[r]=p[k];
p[k]=t;
}
}
else{
break;
}
r++;
}
strcpy(d[num].name,p[k].pname);
d[num].st=i;
d[num].ct=i+p[k].bt;
p[k].wt=d[num].st-p[k].at;
p[k].tt=p[k].wt+p[k].bt;
i=d[num].ct;
p[k].status=1;
k++;
num++;
idle=0;
}
else if(idle==0)
{
strcpy(d[num].name, "idle");
d[num].st=i;
i++;
idle=1;
}
else
{
i++;
}
}
printf (" |pname | arrival time\t | burst time\t | priority\t | status\t | wait\t turn");
for (i =0;i<n; i++)
```

```
{
printf ("\n |%s \t |%d \t \t |%d \t \t |%d \t \t |%d\t \t |%d \t | %d \n", p[i].pname ,p[i].at ,
p[i].bt ,p[i].pr, p[i].status,p[i].wt,p[i].tt);
}
//gant chart
printf("Gant chart\n");
printf("\
n----------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<num;++i){
printf("|%d |\t %s \t ",d[i].st,d[i].name);
}
printf("|%d|",d[num-1].ct);
printf("\
n----------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<n; i++)
{
sumwt=sumwt+p[i].wt;
sumtt=sumtt+p[i].tt;
}
w=(float)(sumwt/n);
st=(float)(sumtt/n);

printf("\nAverage waiting time=%f",w);
printf("\nAverage turnaround time=%f\n", st);
}
```

Output:

```
Enter the number of processes : 3
Enter the process name : p1
Enter the arrival time : 0
Enter the burst time : 5
Enter the priority : 3
Enter the process name : p2
Enter the arrival time : 0
Enter the burst time : 6
Enter the priority : 2
Enter the process name : p3
Enter the arrival time : 0

Enter the burst time : 8
Enter the priority : 0
|pname | arrival time | burst time | priority | status | wait turn
|p3 |0 |8 |0 |1 |0 | 8
|p2 |0 |6 |2 |1 |8 | 14
|p1 |0 |5 |3 |1 |14 | 19
Gant chart
----------------------------------------------------------------------------------------------------------------
--------------------
```

|0 | p3 |8 | p2 |14 | p1 |19|

Average waiting time=7.333333

Average turnaround time=13.666667

RoundRobin Algorithm:
code:


```
#include<stdio.h>
#include<string.h>
int q[100],front=-1,rear=0;
struct process
{
char pname[20];
int at,bt,wt,tt,status,left;
}p[20];
struct done
{
char name[20];
int st,ct;
}d[20];
void enque(int j)
{
q[rear]=j;
rear++;
if (front==-1)
{
front++;
}
}
int deque()
{
int item;
item=q[front];
front++;
if(front==rear)
{
front=-1;
rear=0;
}
return item;
}
void main()
{

int n,i,j,idle=0,k,num,flag=0,found=0,ls,t,nl=0;
float sumwt=0.0,sumtt=0.0,tl,w,st;
num=0;
printf("\nEnter the number of Processes : ");
scanf("%d",&n);
for(i=0;i<n;i++) //accepting the process details
{
printf("\nEnter the process name : ");
__fpurge(stdin);
gets(p[i].pname);
printf("\nEnter the arrival time : ");
```

```
scanf("%d",&p[i].at);
printf("\nEnter the burst time : ");
scanf("%d",&p[i].bt);
p[i].status=0;
p[i].left=p[i].bt;
}
printf("Time Quanta : "); //entering the time slice
scanf("%d",&t);
idle=0;
ls=0;
for(i=0;ls<n;)
{
for(j=0;j<n;j++)
{
if (p[j].at<=i && p[j].status==0)
{
enque(j);
p[j].status=1;
}
}
if (idle==0 && front==-1)
{
strcpy(d[num].name,"idle");
d[num].st=i;
idle=1;
i++;
}
else if(front!=-1)
{
if(idle==1)
{
d[num].ct=i;
idle=0;
num++;
}
k=deque();

d[num].st=i;
strcpy(d[num].name,p[k].pname);
nl++;
if(p[k].left<=t)
{
//printf("\n%s%d",d[num].name,d[num].ct);
d[num].ct=i+p[k].left;
i=d[num].ct;
p[k].tt=i-p[k].at;
p[k].wt=p[k].tt-p[k].bt;
p[k].status=2;
ls++;
num++;
```

```
}
else
{
d[num].ct=i+t;
i=i+t;
p[k].left=p[k].left-t;
num++;
for(j=0;j<n;j++)
{
if(p[j].status==0 && p[j].at<=i)
{
enque(j);
p[j].status=1;
}
}
enque(k);
}
}
else
{
i++;
}
}
printf (" |pname | arrival time\t | burst time\t | status\t | wait\t turn");
for (i =0;i<n; i++)
{
printf ("\n |%s \t |%d \t \t |%d \t \t |%d \t \t |%d \t | %d \n", p[i].pname ,p[i].at , p[i].bt ,
p[i].status,p[i].wt,p[i].tt);
}
//gant chart
printf("Gant chart\n");
printf("\
n-----------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<num;++i){
printf("|%d |\t %s \t ",d[i].st,d[i].name);

}
printf("|%d|",d[num-1].ct);
printf("\
n-----------------------------------------------------------------------------------------------------------------
----------------------\n");
for(i=0;i<n; i++)
{
sumwt=sumwt+p[i].wt;
sumtt=sumtt+p[i].tt;
}
w=(float)(sumwt/n);
st=(float)(sumtt/n);
```

```
printf("\nAverage waiting time=%f",w);
printf("\nAverage turnaround time=%f\n", st);
}
```


output:

```
Enter the number of Processes :
3
Enter the process name : p1
Enter the arrival time : 0
Enter the burst time : 24
Enter the process name : p2
Enter the arrival time : 0
Enter the burst time : 3
Enter the process name : p3
Enter the arrival time : 0
Enter the burst time : 3
Time Quanta : 4
|pname | arrival time | burst time | status | wait turn
|p1 |0 |24 |2 |6 | 30
|p2 |0 |3 |2 |4 | 7
|p3 |0 |3 |2 |7 | 10
Gant chart
-------------------------------------------------------------------------------------------------------------------------------
--------------------
|0 | p1 |4 | p2 |7 | p3 |10 | p1 |14 | p1 |18 | p1 |22 | p1
|26 | p1 |30|

-------------------------------------------------------------------------------------------------------------------------------
--------------------
Average waiting time=5.666667
Average turnaround time=15.666667
```

Producer and Consumer

CODE:


```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#include<time.h>
sem_t mutex,empty,full;
int buffer[5],get=0,gitem,item=0,put=0,pro[20],con[20];
void *producer(void *arg)
{
do
{
sem_wait(&empty);
sem_wait(&mutex);
buffer[put%5]=item;
item++;
printf("producer %d produces %d item buffered[%d]:%d\n",(*(int *)arg),buffer[put%5],put
%5,item);
put++;
sem_post(&mutex);
sem_post(&full);
sleep(3);
}
while(1);
}
void *consumer(void *arg)
{
do
{
sem_wait(&full);
sem_wait(&mutex);
gitem=buffer[get%5];
printf("consumer %d consumes %d item buffered[%d]:%d\n",(*(int *)arg),gitem,get%5,gitem);
get++;
sem_post(&mutex);
sem_post(&empty);
sleep(2);
}
while(1);
}
void main()
{
int p,c,j,k;
pthread_t a[10],b[10];
sem_init(&mutex,0,1);
sem_init(&full,0,0);
sem_init(&empty,0,5);
printf("\n enter the no of processes");
scanf("%d",&p);
printf("enter the no of consumers");
```

```
scanf("%d",&c);for(j=0;j<p;j++)
{
pro[j]=j;
pthread_create(&a[j],NULL,producer,&pro[j]);
}
for(k=0;k<c;k++)
{
con[k]=k;
pthread_create(&b[k],NULL,consumer,&con[k]);
}
for(j=0;j<p;j++)
{
pthread_join(a[j],NULL);
}
for(k=0;k<c;k++)
{
pthread_join(b[k],NULL);
}
}
```

OUTPUT:

```
 enter the no of processes 4
enter the no of consumers 3
producer 0 produces 0 item buffered[0]:1
producer 1 produces 1 item buffered[1]:2
producer 3 produces 2 item buffered[2]:3
consumer 1 consumes 0 item buffered[0]:0
producer 2 produces 3 item buffered[3]:4
consumer 0 consumes 1 item buffered[1]:1
consumer 2 consumes 2 item buffered[2]:2
consumer 1 consumes 3 item buffered[3]:3
producer 1 produces 4 item buffered[4]:5
producer 3 produces 5 item buffered[0]:6
consumer 2 consumes 4 item buffered[4]:4
producer 2 produces 6 item buffered[1]:7
consumer 0 consumes 5 item buffered[0]:5
producer 0 produces 7 item buffered[2]:8
consumer 1 consumes 6 item buffered[1]:6
^C
```

PAGE REPLACMENT:

FIFO

CODE:

```
#include<stdio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1;
void main(){
printf("\nEnter no of frame..");
scanf("%d",&nof);
printf("\nEnter no of pages");
scanf("%d",&nor);
printf("\nEnter the page nos");
for(i=0;i<nor;i++){
scanf("%d",&ref[i]);}
printf("\n The given page no's are");
for(i=0;i<nor;i++){printf("%4d",ref[i]);}
for(i=1;i<=nof;i++){
frm[i]=-1;}
for(i=0;i<nor;i++){
flag=0;
printf("\n\t page no %d-> ",ref[i]);
for(j=0;j<nof;j++){
if(frm[j]==ref[i]){
flag=1;break;}}
if(flag==0){
pf++;
victim++;
victim=victim%nof;
frm[victim]=ref[i];
for(j=0;j<nof;j++)
printf("%4d",frm[j]);}}
printf("\n\n\n No.of pages faults %d",pf);}
```

OUTPUT:

```
 FIFO PAGE REPLACEMENT
Enter number of frames:4
Enter the number of pages:6
Enter the page numbers:5 6 4 1 2 3

 the given pages are:   5   6   4   1   2   3

        page no 5->       5  -1  -1  -1
        page no 6->       5   6  -1  -1
        page no 4->       5   6   4  -1
        page no 1->       5   6   4   1
        page no 2->       2   6   4   1
        page no 3->       2   3   4   1
        number of page faults:6
```

LRU Page replacment :

CODE:


```
#include<stdio.h>
void main()
{
int i, j ,k, min, rs[25], m[10], count[10], flag[25]={0}, n, f, pf=0, next=1;
printf("Enter the length of reference string : ");
scanf("%d" ,&n);
printf("Enter the reference string : ");
for(i=0;i<n;i++)
{
scanf("%d", &rs[i]);
}
printf("Enter the number of frames : ");
scanf("%d" ,&f);
for(i=0;i<f;i++)
{
count[i]=0;
m[i]=-1;
}
printf("\n\nThe Page Replacement process is : \n");
for(i=0;i<n; i++)
{
for(j=0; j<f; j++)
{
if(m[j]==rs[i])
{
flag[i]=1;
count[j]=next;

next++;
}
}
if (flag[i]==0)
{
if(i<f)
{
m[i]=rs[i];
count[i]=next;
next++;
}
else
{
min=0;
for(j=1; j<f; j++)
if(count[min] > count[j])
min=j;
m[min]=rs[i];
count[min]=next;
next++;
```

```
}
pf++;
}
for (j=0; j<f; j++)
printf("%d\t", m[j]);
if (flag[i]==0)
printf("PF No. : %d" , pf);
printf("\n");
}

printf("\nThe number of page faults using LRU are : %d\n",pf);
}
```

output:

```
enter the length of reference string*-- 20
enter the reference string--1
4
5
3
2
1
2
3
4
5
4
3
455
6
7
6
54
6
3
4
enter the no of frames--2

the replacement process is
1       -1      PF no. --1
1       4       PF no. --2
5       4       PF no. --3
5       3       PF no. --4
2       3       PF no. --5
2       1       PF no. --6
2       1
2       3       PF no. --7
4       3       PF no. --8
4       5       PF no. --9
4       5
4       3       PF no. --10
455     3       PF no. --11
455     6       PF no. --12
```

```
7        6        PF no. --13
7        6
54       6        PF no. --14
54       6
8        6        PF no. --15
8        4        PF no. --16

the number of faults using LRU are 16
```

FCFS DISK SHEDULING:

CODE:

```
#include<stdio.h>
#include<stdlib.h>
void main(){
int t[20],n,i,j,tot=0;
float avhm;
printf("enter the no of tracks:");
scanf("%d",&n);
printf("enter the current position:");
scanf("%d",&t[0]);
printf("enter the tracks to be travrsed");
for(i=1;i<n+1;++i)
scanf("%d",&t[i]);
for(i=0;i<n;++i){
tot=tot+abs(t[i]-t[i+1]);
}
printf("\nTotal head movement : %d\n",tot);
avhm=(float)tot/n;
printf("\nAverage head movement: %f",avhm);
}
```

Output:

```
enter the number of tracks:8
enter the current position:53
enter the tracks to be traversed:98 183 37 122 14 124 65 67

total head movements:640

average head movements:80.000000
```

SCAN:

CODE:


```
#include<stdio.h>
#include<stdlib.h>
main()
{
int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0,end=199;
printf("Enter the no of tracks to be traversed : ");
scanf("%d" ,&n);
printf("Enter the position of head : ");
scanf("%d" ,&h);
t[0]=0;t[1]=h;
t[n+2]=end;
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]=temp;
}
}
}
for(i=0;i<n+2; i++)
if(t[i]==h)
{
k=i;
p=0;
}
if(h<(end-h))
{
for(i=k;i>=0;i--)
{ atr[p]=t[i];
p++;
}
for(i=k+1;i<n+2;i++)
{
atr[p]=t[i];
p++;
}
}
else

{
for (i=k;i<=n+2;i++)
{
```

```
atr[p]=t[i];
p++;
}
for(i=k-1;i>0;i--)
{
atr[p]=t[i];
p++;
}
}
printf("Scheduling order : \n");
for (p=0;p<n+2;p++)
printf("%d \t",atr[p]);
for (j=0; j<n+1; j++)
{sum=sum+abs(atr[j]-atr[j+1]);
}
printf("\nTotal head movements:%d\n",sum);
}
```

output:

```
enter the number of tracks to be travered 8
enter the position of head 53
enter the tracks 98 183 37 122 14 124 65 67
schduling order:
53      37      14      0       65      67      98      122     124     183
Total head movments:236
```

CSCAN:

CODE:


```
#include<stdio.h>
main()
{
int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0,end=199;
printf("Enter the no of tracks to be traversed : ");
scanf("%d" ,&n);
printf("Enter the position of head : ");
scanf("%d" ,&h);
t[0]=0;t[1]=h;
t[n+2]=end;
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]=temp;
}
}
}
for(i=0;i<n+2; i++)
if(t[i]==h)
{
k=i;
p=0;
}
if(h<(end-h))
{
for(i=k;i>=0;i--)
{ atr[p]=t[i];
p++;
}
for(i=n+2;i>k;i--)
{
atr[p]=t[i];
p++;
}
}
else

{
for (i=k;i<=n+2;i++)
{
atr[p]=t[i];
```

```
p++;
}
for(i=0;i<k;i++)
{
atr[p]=t[i];
p++;
}
}
printf("Scheduling order : \n");
for (p=0;p<n+2;p++)
printf("%d \t",atr[p]);
for (j=0; j<n+1; j++)
{sum=sum+abs(atr[j]-atr[j+1]);
}
printf("\nTotal head movements:%d\n",sum);
}
```

output;

```
Enter the number of tracks to be traversed: 8
Enter the postion of head: 53
Enter the tracks: 3 4 98 76 54 180 100 3 1 7 8
Scheduling order:
53      4       3       3       0       199     180     100     98      76
Total head movements: 397
```

BANKERS:

CODE:


```
#include<stdio.h>
struct file{
int allo[10],max[10],need[10],flag;
}f[10];
int main(){
int i=0,n,m;
int safe[10];
int avail[10],req[10];
printf("enter the no of process:");
scanf("%d",&n);
printf("enter the no of resourses:");
scanf("%d",&m);
for(i=0;i<n;++i){
printf("Enter the details of P%d ",i);
printf("Enter the allocation details");
for(int j=0;j<m;++j)
scanf("%d",&f[i].allo[j]);
printf("Enter the maximum requirement:");
for(int j=0;j<m;++j)
scanf("%d",&f[i].max[j]);
f[i].flag=0;
}
for(i=0;i<n;i++){
for(int j=0;j<m;++j){
f[i].need[j]=f[i].max[j]-f[i].allo[j];
}
}
printf("enter the currently available resource");
for(i=0;i<m;++i)
scanf("%d",&avail[i]);
printf("Enter the new request details\n enter pid:");
int pid=-1;
scanf("%d",&pid);
printf("Enter the request for resources");
for(i=0;i<m;++i)
scanf("%d",&req[i]);
//checking the request
int check=0;
for(int j=0;j<m;j++){
if(f[pid].need[j]<req[j] && avail[j]<req[j] )
check=1;

}
int safeid=0;
if(check==1)

{printf("the request cannot be permitted");return 0;}
else{
```

```
for(i=0;i<m;i++){
avail[i]=avail[i]+f[pid].allo[i];
f[pid].need[i]=f[pid].need[i]-req[i];
f[pid].allo[i]=f[pid].allo[i]+req[i];
f[pid].flag=1;
//printf("need:%d alloc: %d",f[pid].need[i],f[pid].allo[i]);
}
safe[0]=pid; safeid=1;
}
int accpt;
for(i=0;safeid<n;i=(i+1)%n){
accpt=0;
for(int j=0;j<m;j++){
if(f[i].need[j]>avail[j]){accpt=1;break;}
}
if(accpt==0 && f[i].flag==0){
safe[safeid]=i;safeid++;
for(int j=0;j<m;j++)
avail[j]+=f[i].allo[j];
f[i].flag=1;

}
}
printf("\nPID\t max \t alloc \t need\n");
for(i=0;i<n;++i){
printf("P%d\t",i);
for(int j=0;j<m;j++)printf("%d ",f[i].max[j]);
printf("\t");
for(int j=0;j<m;j++)printf("%d",f[i].allo[j]);
printf("\t");
for(int j=0;j<m;j++)printf("%d",f[i].need[j]);
printf("\n");
}
printf("\n");
for(int j=0;j<n;j++)
printf(" P%d ",safe[j]);
return 0;
}
```

OUTPUT:


```
Enter the details for P0
Enter allocation         --     0 1 0
Enter Max                --     7 5 3
Enter the details for P1
Enter allocation         --     2 0 0
Enter Max                --     3 2 2
Enter the details for P2
Enter allocation         --     3 0 2
Enter Max                --     9 0 2
Enter the details for P3
Enter allocation         --     2 1 1
Enter Max                --     2 2 2
Enter the details for P4
Enter allocation         --     0 0 2
Enter Max                --     4 3 3

Enter Available Resources        --     3 3 2

Enter new request details --
Enter pid        --     1
Enter Request for Resources      --     1 0 2

P1 is visited(  5  3  2)
P3 is visited(  7  4  3)
P4 is visited(  7  4  5)
P0 is visited(  7  5  5)
P2 is visited( 10  5  7)
System is in safe state
The safe sequence is -- (P1 P3 P4 P0 P2 )
Process         Allocation              Max                     Need
P0          0     1     0     7     5     3     7     4     3
P1          3     0     2     3     2     2     0     2     0
P2          3     0     2     9     0     2     6     0     0
P3          2     1     1     2     2     2     0     1     1
P4          0     0     2     4     3     3     4     3     1
```

FIRST FIT MEMORY ALLOCATION:

CODE:

```
#include<stdio.h>
void main(){
int block[10],proc[2][10];
int i,j,bloNo,proNo;
printf("Enter the no of blocks");
scanf("%d",&bloNo);
printf("Enter the no of process");
scanf("%d",&proNo);
printf("Enter the size of the blocks");
i=0;
while(i<bloNo){scanf("%d",&block[i]);i++;}
i=0;
printf("Enter the size of each process");
for(i=0;i<proNo;++i){
scanf("%d",&proc[0][i]);
}
i=0;
while(i<proNo){
j=0;
while(j<bloNo){
if(proc[0][i]<block[j]){
proc[1][i]=j+1;
block[j]-=proc[0][i];
break;
}
j++;
}
if(j==bloNo)
{proc[1][i]=-1;}
i++;
}
printf("OUTPUT\n");
i=0;
printf("pr.NO \t process size\t Block NO\n");
while(i<proNo){
printf("%d",i+1);
printf("\t %d\t ",proc[0][i]);
if(proc[1][i]==-1){
printf("\tNot allocated");
}
else
printf("\t%d",proc[1][i]);
printf("\n");
i++;
}}
```

OUTPUT:
Enter the no of blocks5
Enter the no of process4
Enter the size of the blocks10 15 5 9 3
Enter the size of each process1 4 7 12

OUTPUT

pr.NO	process size	Block NO
1	1	1
2	4	1
3	7	2
4	12	Not allocated

BEST FIT MEMORY ALLOCATION

CODE:

```
#include<stdio.h>
void main(){
int block[10],proc[2][10];
int i,j,bloNo,proNo;
printf("Enter the no of blocks");
scanf("%d",&bloNo);
printf("Enter the no of process");
scanf("%d",&proNo);
printf("Enter the size of the blocks");
i=0;
while(i<bloNo){scanf("%d",&block[i]);i++;}
i=0;
printf("Enter the size of each process");
for(i=0;i<proNo;++i){
scanf("%d",&proc[0][i]);
}
i=0;
while(i<proNo){
j=0;
int index=-1,flag=0;
while(j<bloNo){
if(proc[0][i]<=block[j]){
if(flag==0){
index=j;flag=1;
}
else{
if(block[j]<block[index])
index=j;

}
}
j++;
}
proc[1][i]=index+1;
block[index]-=proc[0][i];
i++;
}
printf("OUTPUT\n");
i=0;
printf("pr.NO \t process size\t Block NO\n");
while(i<proNo){
printf("%d",i+1);
printf("\t %d\t ",proc[0][i]);
if(proc[1][i]==0){
printf("\tNot allocated");
}
else
```

```
printf("\t%d",proc[1][i]);

printf("\n");
i++;
}
}
```


OUTPUT:
Enter the no of blocks5
Enter the no of process4
Enter the size of the blocks10 15 5 9 3
Enter the size of each process1 4 7 12

OUTPUT

pr.NO	process size	Block NO
1	1	5
2	4	3
3	7	4
4	12	2

WORST FIT MEMORY ALLOCATION

CODE:

```
#include<stdio.h>
void main(){
int block[10],proc[2][10];
int i,j,bloNo,proNo;
printf("Enter the no of blocks");
scanf("%d",&bloNo);
printf("Enter the no of process");
scanf("%d",&proNo);
printf("Enter the size of the blocks");
i=0;
while(i<bloNo){scanf("%d",&block[i]);i++;}
i=0;
printf("Enter the size of each process");
for(i=0;i<proNo;++i){
scanf("%d",&proc[0][i]);
}
i=0;
while(i<proNo){
j=0;
int index=-1,flag=0;
while(j<bloNo){
if(proc[0][i]<=block[j]){
if(flag==0){
index=j;flag=1;
}
else{
if(block[j]>block[index])
index=j;

}
}
j++;
}
proc[1][i]=index+1;
block[index]-=proc[0][i];
i++;
}
printf("OUTPUT\n");
i=0;
printf("pr.NO \t process size\t Block NO\n");
while(i<proNo){
printf("%d",i+1);
printf("\t %d\t ",proc[0][i]);
if(proc[1][i]==0){
printf("\tNot allocated");
}
else
```

```
printf("\t%d",proc[1][i]);

printf("\n");
i++;
}
}
```

OUTPUT:
Enter the no of blocks5
Enter the no of process4
Enter the size of the blocks10 15 5 9 3
Enter the size of each process1 4 7 12
OUTPUT

pr.NO	process size	Block NO
1	1	2
2	4	2
3	7	1
4	12	Not allocated