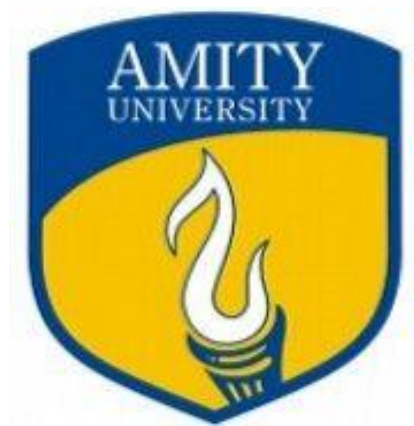


Operating System (CSE202)
LAB FILE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Submitted to:
Ms. Seema Sharma

Submitted by:
Neeyati Shivind Jauhar
A2305210290
4CSE-4Y

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY UTTAR PRADESH
NOIDA – 201301

INDEX

No.	Assignment Category	Code	Experiment No.	Name of Experiment	Date of Allotment of Experiment	Date of Evaluation	Max. Marks	Marks Obtained	Signature of Faculty
1.	Mandatory Experiment	LR (10)	1	To explore basic Linux commands	17/12/20		1		
2.			2	To explore basic Linux commands	24/12/20		1		
3.			3	To explore Linux file commands	07/01/21		1		
4.			4	To explore Linux file commands	14//01/21		1		
5.			5	Programs based on Shell Scripting	21/01/21		1		
6.			6	Programs based on Shell Scripting	04/02/21		1		
7.			7	Programs based on Shell Scripting	11/02/21		1		
8.			8	Program to simulate First Come First Serve CPU scheduling algorithm	18/02/21		1		
9.			9	Program to simulate Shortest Job First CPU scheduling algorithm	04/03/21		1		
10.			10	Write a program to implement round robin scheduling algorithm.	18/03/21		1		
11.			11	Write a program to implement Banker's Algorithm.	18/03/21		1		
12.			12	Write a program to implement LOOK disc scheduling algorithm.	25/03/21		1		
	Viva		Viva						

Experiment – 1

Objective: To explore basic Linux commands.

Software used: Cygwin64 Terminal

Theory:

- 1) man: It is used to display the user manual of any command
Syntax: man <command>

```
HP@neeyati-pc ~
$ man echo

HP@neeyati-pc ~
$ |
ECHO(1)                                     User Commands

NAME
    echo - display a line of text

SYNOPSIS
    echo [SHORT-OPTION]... [STRING]...
    echo LONG-OPTION

DESCRIPTION
    Echo the STRING(s) to standard output.

    -n      do not output the trailing newline
    -e      enable interpretation of backslash escapes
    -E      disable interpretation of backslash escapes (default)
    --help  display this help and exit
    --version
            output version information and exit

    If -e is in effect, the following sequences are recognized:

    \\      backslash
    \a      alert (BEL)
    \b      backspace
    \c      produce no further output
    \e      escape
    \f      form feed
    \n      new line
    \r      carriage return
    \t      horizontal tab
    \v      vertical tab
    \ONNN   byte with octal value NNN (1 to 3 digits)
    \xHH    byte with hexadecimal value HH (1 to 2 digits)

    NOTE: your shell may have its own version of echo, which usually supersedes the version described here. Please refer to your shell's documentation for

AUTHOR
    Written by Brian Fox and Chet Ramey.

REPORTING BUGS
    GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
    Report echo translation bugs to <http://translationproject.org/team/>

SEE ALSO
    Full documentation at: <http://www.gnu.org/software/coreutils/echo>
    or available locally via: info '(coreutils) echo invocation'
```

- 2) echo: It is used to display line of text/string that are passed into the argument.
Syntax: echo <string to be displayed>

```
HP@neeyati-pc ~
$ echo Hello World
Hello World
```

- 3) help: it is used to display information about the command from the shell's documents.
Syntax: run-help <command> or <command> --help

```
HP@neeyati-pc ~
$ help
GNU bash, version 4.4.12(3)-release (x86_64-unknown-cygwin)
These shell commands are defined internally.  Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f filename] [-q name] [-u name] [-r keyseq] [-x keyseq:shell-command] [keyseq:>
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...] COMMANDS ;;)... esac
cd [-L|[-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abdefgjklsuv] [-o option] [-A action] [-G globpat] [-W wordlist] [-F function] [-C command] [-X fi>
complete [-abdefgjklsuv] [-pr] [-DE] [-o option] [-A action] [-G globpat] [-W wordlist] [-F function] [-C co>
comptop [-o|+o option] [-DE] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFgIlNrtux] [-p] [name[=value] ...]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid ...]
echo [-neE] [arg ...]
enable [-a] [-dnps] [-f filename] [name ...]
eval [arg ...]
exec [-cl] [-a name] [command [arguments ...]] [redirection ...]
exit [n]
export [-fn] [name[=value] ...] or export -p
false
fc [-e ename] [-lnr] [first] [last] or fc -s [pat=rep] [command]
fg [job_spec]
for NAME [in WORDS ... ] ; do COMMANDS; done
for (( exp1; exp2; exp3 )); do COMMANDS; done
function name { COMMANDS ; } or name () { COMMANDS ; }
getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt] [name ...]
help [-dms] [pattern ...]
```

- 4) clear: it is used to clear the terminal screen
Syntax: clear

```
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFgIlNrtux] [-p] [name[=value] ...]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid ...]
echo [-neE] [arg ...]
enable [-a] [-dnps] [-f filename] [name ...]
eval [arg ...]
exec [-cl] [-a name] [command [arguments ...]] [redirection ...]
exit [n]
export [-fn] [name[=value] ...] or export -p
false
fc [-e ename] [-lnr] [first] [last] or fc -s [pat=rep] [command]
fg [job_spec]
for NAME [in WORDS ... ] ; do COMMANDS; done
for (( exp1; exp2; exp3 )); do COMMANDS; done
function name { COMMANDS ; } or name () { COMMANDS ; }
getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt] [name ...]
help [-dms] [pattern ...]

HP@neeyati-pc ~
$ clear
```

```
HP@neeyati-pc ~
$
```

- 5) alias: it is used to create custom shortcuts to represent commands.

Syntax: alias <aliasName>=<command>

```
HP@neeyati-pc ~  
$ alias c=clear
```

- 6) uname: it is used to display the basic information about the operating system name and system hardware.

Syntax: uname <options>

```
HP@neeyati-pc ~  
$ uname  
CYGWIN_NT-10.0  
  
HP@neeyati-pc ~  
$
```

- 7) whoami: it is used to display the username of the current user when the command is invoked.

Syntax: whoami

```
HP@neeyati-pc ~  
$ whoami  
HP
```

- 8) pwd: it is used to display the current working directory.

Syntax: pwd

```
HP@neeyati-pc ~  
$ pwd  
/home/HP
```

- 9) history: it is used to display a list of commands that have been recently used.

Syntax: history

```
HP@neeyati-pc ~  
$ history  
1  --help  
2  echo --help  
3  ls  
4  who  
5  cat >file1  
6  cat file1  
7  cat > file2  
8  cat file1 > file2  
9  cat file1 file2  
10 cat file1 file2 > file3  
11 cat file3  
12 cat > file2  
13 cat file1 file2 > file3  
14 cat file3  
15 clear  
16 who  
17 echo "SHELL"  
18 echo "$SHELL"  
19 clear  
20 man  
21 help  
22 pwd  
23 clear  
24 echo "$SHELL"  
25 man echo  
26 alias c=clear  
27 who  
28 history  
  
HP@neeyati-pc ~  
$
```

10) cat: it is used to create single or multiple files, view content of file/s, concatenate files and redirect output in terminal or files.

Syntax: cat > filename	(To create a new file)
cat filename	(To open a file)
cat >> filename	(To append the content of a file)
cat file1>file2	(To copy content of file 1 into file 2)

```
HP@neeyati-pc ~  
$ cat > file1  
file1 Hi. I am Neeyati.  
  
HP@neeyati-pc ~  
$ cat > file2  
file2 I am 19 years old.  
  
HP@neeyati-pc ~  
$ cat file1 file2 > file3  
  
HP@neeyati-pc ~  
$ cat file3  
file1 Hi. I am Neeyati.  
file2 I am 19 years old.  
  
HP@neeyati-pc ~  
$
```

Result: Successfully executed basic Linux commands.

Experiment – 2

Objective: To explore Linux file commands.

Software used: Cygwin64 Terminal

Theory:

- 11) cd: It is used to change the current working directory
Syntax: cd <new_directory>
- 12) ls: It is used to list the files in the current directory
Syntax: ls <options>
- 13) mkdir: it is used to create a new directory/folder
Syntax: mkdir <directory_name>
rmdir – used to remove empty directory
- 14) mv: it is used to move or rename a file/directory
Syntax: mv <file_name> <new_directory>
- 15) cp: it is used to copy file or directory.
Syntax: cp <source_file> <dest_file>
cp -r – used to copy directory
cp -i – used to avoid overwriting pre-existing file/directory
- 16) ln: it is used to make links (soft and hard)
Syntax: ln <file1> <file2> to create hard link
ln -s <file1> <file2> to create soft link
- 17) rm: it is used to remove a file or a directory
Syntax: rm <filename>

Command Execution :

```
E ~
HP@neeyati-pc ~
$ mkdir new1
HP@neeyati-pc ~
$ ls
file1 file2 file3 new new1

HP@neeyati-pc ~
$ ls -lt
total 3
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:32 new1
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:29 new
-rw-r--r-- 1 HP HP 49 Dec 17 22:29 file3
-rw-r--r-- 1 HP HP 25 Dec 17 22:28 file2
-rw-r--r-- 1 HP HP 24 Dec 17 22:28 file1

HP@neeyati-pc ~
$ cp file2 file1

HP@neeyati-pc ~
$ mv file3 file1
```

```
HP@neeyati-pc ~  
$ mv new1 new  
  
HP@neeyati-pc ~  
$ ls  
file1 file2 file3 new  
  
HP@neeyati-pc ~  
$ cd new  
  
HP@neeyati-pc ~/new  
$ ls  
file1 new1 new2  
  
HP@neeyati-pc ~/new  
$ mkdir os1  
  
HP@neeyati-pc ~/new  
$ ls  
file1 new1 new2 os1
```

Result: Successfully executed Linux file commands.

Experiment – 3

Objective: To explore Linux file commands.

Software used: Cygwin64 Terminal

Theory:

18) cmp: It is used to compare the files byte by byte.

Syntax: cmp <option> <file1> <file2>

cmp -b: it also displays the character where the first mismatch occurred

cmp -i <num>: bytes to be skipped in both the files

cmp -n <num>: number of bytes to be compared

```
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat > file1
File
to
compare
^C
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat > file2
File
to
compared
^C
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cmp file1 file2
file1 file2 differ: char 17, line 3
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cmp -b file1 file2
file1 file2 differ: byte 17, line 3 is  12 ^J 144 d
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cmp -i 4 file1 file2
file1 file2 differ: char 13, line 3
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cmp -n 10 file1 file2
```

19) sort: It is used to sort the file in alphabetical order only for display.

Syntax: sort <options> <filename>

sort -r: to sort in reverse alphabetical order

sort -n: to sort from lowest to highest number

sort -u: to remove duplicates and sort

sort -m: to sort on the basis of months

sort -f: to ignore case sensitivity and sort

```
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat > colour1
Black
Pink
Yellow
Red
green^C
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort colour1
Black
Pink
Red
Yellow
green
[rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort -f colour1
Black
green
Pink
Red
Yellow
```

```

rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort -rf colour1
Yellow
Red
Pink
green
Black
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat > months
December
March
June
April
^C
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort -M months
March
April
June
December
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat > number
7
4
2
9
2
^C
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort -n number
2
2
4
7
9
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort -u number
2
4
7
9

```

To sort and store: sort <src_filename> > <dest_filename>

```

rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort colour1 > sortColour1
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % sort colour2 > sortColour2
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat colour1 colour2
Black
Pink
Yellow
Red
greenBlue
Orange
green
Red
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat sortColour1 sortColour2
Black
Pink
Red
Yellow
green
Blue
Orange
Red
green

```

20) comm: it is used to find the unique and common words in different files

Syntax: comm <options> <file1> <file2>

comm -12 <file1> <file2>: will only show common words in both files and remove the columns 1 and 2.

```
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % comm sortColour1 sortColour2
Black
      Blue
      Orange
Pink
      Red
Yellow
      green
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % comm -12 sortColour1 sortColour2
Red
green
```

21) wc: it is used to show the line, word and character count in the file.

Syntax: wc <options> <filename>

wc -l: to show only the line count

wc -w: to show only the word count

wc -m: to show only the character count

```
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % cat colour1
Black
Pink
Yellow
Red
green
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % wc colour1
  4      5     27 colour1
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % wc -l colour1
  4 colour1
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % wc -m colour1
 27 colour1
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % wc -w colour1
  5 colour1
```

22) vi: it is used to open the Visual editor. It is the default editor that comes with the UNIX operating system. It has **three** modes:

- i. Command mode, where commands which take action on the file are written. Opens by default as we enter the vi.
- ii. Insert mode, where the entered text is inserted into the file. Opens by using commands such as 'i', 'a', 'A', 'o', etc.
- iii. Escape mode, from where the file can be saved and editor can be quitted. Opens by using the command ':'.

Syntax: vi <filename>

```
rajeevseghal@RAJEEVs-MacBook-Air Folder1 % vi editor
```

December
March
June
April

2

2

2

2

10

1

2

2

2

2

2

2

1

3

2

2

2

2

2

1992

1

2

```
:wq months
```

Result: Successfully executed Linux file commands.

Experiment – 4

Objective: To explore Linux file commands.

Software used: Cygwin64 Terminal

Theory:

23) chmod: It is used to change the access permission for the files or directories.

Syntax: chmod <class/es> <option> <permission/s> <file_name/dir_name>

chmod <numeric_value> <file_name/dir_name>

```
HP@neeyati-pc ~
$ chmod ugo+w fileos

HP@neeyati-pc ~
$ ls -l
total 3
-rw-r--r-- 1 HP HP 98 Jan  7 12:35 file1
-rw-r--r-- 1 HP HP 25 Dec 17 22:28 file2
drwxr-xr-x+ 1 HP HP  0 Dec 24 12:54 file3
-rw-rw-rw- 1 HP HP 18 Jan 14 11:59 fileos
drwxr-xr-x+ 1 HP HP  0 Dec 24 12:56 new

HP@neeyati-pc ~
$ cat >>fileos
second line

HP@neeyati-pc ~
$ mkdir fileos1

HP@neeyati-pc ~
$ ls
file1 file2 file3 fileos fileos1 new

HP@neeyati-pc ~
$ cd ..
```

24) ping: It is used to check network connectivity between host and server/host.

Syntax: ping <option> <IP/URL>

25) ps: it is used to display information related to the processes running in the system.

Syntax: ps <options>

```
HP@neeyati-pc /
$ ps
  PID   PPID   PGID   WINPID   TTY      UID    STIME  COMMAND
  1494   1493   1494   13740    pty0     197609 11:58:45 /usr/bin/bash
  1519   1494   1519    9872    pty0     197609 12:12:47 /usr/bin/ps
  1493     1   1493   14040    ?        197609 11:58:45 /usr/bin/mintty
```

26) tty: it is used to display the file name of the terminal connected to standard input.

Syntax: wc <options>

- 27) **ls more** : lets you view text files or other output in a scrollable manner
ls less : can be used to read contents of text file one page(one screen) per time

```
HP@neeyati-pc ~  
$ ls -l | more  
total 3  
-rw-r--r-- 1 HP HP 98 Jan 7 12:35 file1  
-rw-r--r-- 1 HP HP 25 Dec 17 22:28 file2  
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:54 file3  
-rw-rw-rw- 1 HP HP 31 Jan 14 12:01 fileos  
drwxr-xr-x+ 1 HP HP 0 Jan 14 12:09 fileos1  
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:56 new  
  
HP@neeyati-pc ~  
$ ls -l | less  
total 3  
-rw-r--r-- 1 HP HP 98 Jan 7 12:35 file1  
-rw-r--r-- 1 HP HP 25 Dec 17 22:28 file2  
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:54 file3  
-rw-rw-rw- 1 HP HP 31 Jan 14 12:01 fileos  
drwxr-xr-x+ 1 HP HP 0 Jan 14 12:09 fileos1  
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:56 new  
(END)
```

- 28) **Pipe**: it is used to combine two or more commands, the output of one command acts as the input to another command.

Syntax: <command> | <command> | <command>

```
HP@neeyati-pc ~  
$ ls -l | more  
total 3  
-rw-r--r-- 1 HP HP 98 Jan 7 12:35 file1  
-rw-r--r-- 1 HP HP 25 Dec 17 22:28 file2  
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:54 file3  
-r-xr-x--x 1 HP HP 63 Jan 14 12:32 fileos  
drwxr-xr-x+ 1 HP HP 0 Jan 14 12:09 fileos1  
drwxr-xr-x+ 1 HP HP 0 Dec 24 12:56 new
```

- 29) **grep** – it is used to search a file for a particular pattern of characters and displays all the lines that contain that pattern.

Syntax – **grep** <option> <keyword/pattern> <filename>

grep -i: ignores case sensitivity

grep -n: displays pattern along with line number

grep -v: displays lines not matching

grep -c: displays matching lines

grep -e: used to search for multiple patterns

```
HP@neeyati-pc ~  
$ cat >> fileos  
car  
car1  
truck  
CAR  
bus  
cycle  
  
HP@neeyati-pc ~  
$ grep -in car fileos  
3:car  
4:car1  
6:CAR  
  
HP@neeyati-pc ~  
$ grep -c car fileos  
2  
  
HP@neeyati-pc ~  
$ grep -ic car fileos  
3
```

Result: Successfully executed Linux file commands.

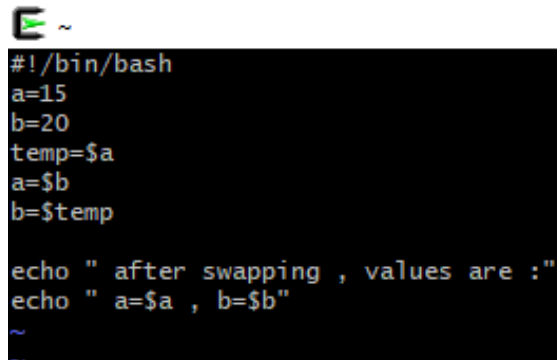
Experiment – 5

Objective: To write scripts for given questions

Software used: Cygwin64 Terminal

Q1. Write a script to swap 2 numbers.

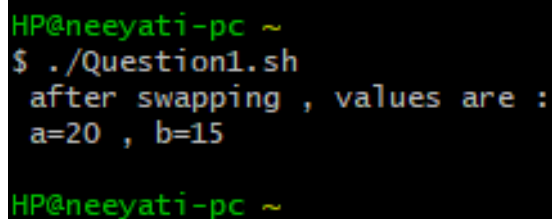
Code:



```
#!/bin/bash
a=15
b=20
temp=$a
a=$b
b=$temp

echo " after swapping , values are :"
echo " a=$a , b=$b"
~
~
```

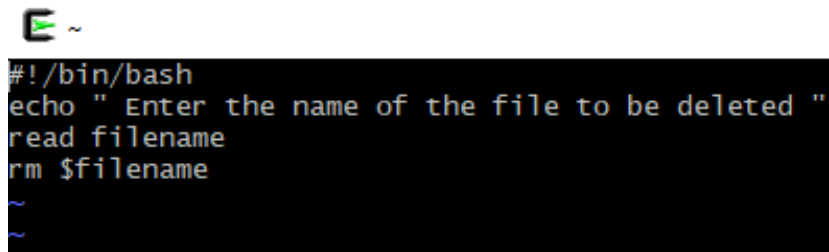
Output:



```
HP@neeyati-pc ~
$ ./Question1.sh
 after swapping , values are :
a=20 , b=15
HP@neeyati-pc ~
```

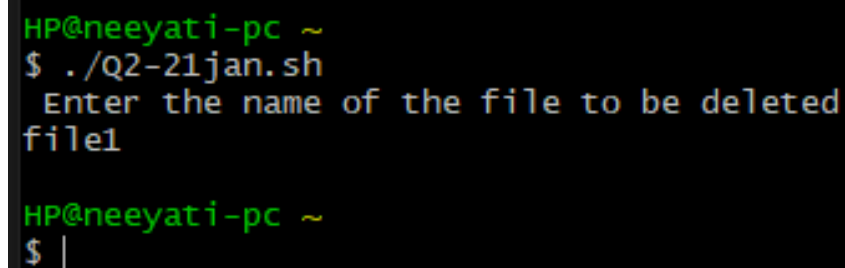
Q2. Write a script to delete file.

Code:



```
#!/bin/bash
echo " Enter the name of the file to be deleted "
read filename
rm $filename
~
~
```

Output:



```
HP@neeyati-pc ~
$ ./Q2-21jan.sh
 Enter the name of the file to be deleted
file1
HP@neeyati-pc ~
$ |
```

Q3. Write script to add text to already existing file.

Code:

```
HP~  
#!/bin/bash  
echo " enter text to be added "  
read text  
echo $text >>addText  
~  
~
```

Output:

```
HP@neeyati-pc ~  
$ chmod 777 question2.sh  
HP@neeyati-pc ~  
$ ./Question2.sh  
enter text to be added  
hi  
HP@neeyati-pc ~  
$
```

Q4. Write a script to demonstrate the use of arithmetic operators.

Code:

```
HP~  
#!/bin/bash  
a=10  
b=20  
val=`expr $a + $b`  
echo " a + b = $val"  
  
val=`expr $a "*" $b`  
echo " a*b = $val"  
~  
~
```

Output:

```
HP@neeyati-pc ~  
$ ./Question3.sh  
a + b = 30  
a*b = 200  
HP@neeyati-pc ~  
$ |
```


Experiment – 6

Objective: To write scripts for given questions

Software used: Cygwin64 Terminal

Q1. Write a script to check whether number is positive or negative.

Code:

```
E ~
echo " enter a number "
read num
if [ $num -lt 0 ]
then
echo "NEGATIVE NUMBER"
elif [ $num -gt 0 ]
then
echo "POSITIVE NUMBER"
else
echo " Number is 0"
fi
~
```

Output:

```
HP@neeyati-pc ~
$ ./Q1-4feb.sh
enter a number
3
POSITIVE NUMBER
```

Q2. Write a script to find greatest number among three numbers.

Code:

```
E ~ - □ ×
echo " enter num1 "
read num1
echo " enter num2"
read num2
echo " enter num3"
read num3
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
echo " $num1 is greatest"
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
echo " $num2 is greatest"
else
echo " $num3 is greatest"
fi
~
```

Output:

```
HP@neeyati-pc ~
$ chmod 777 Q2-4feb.sh

HP@neeyati-pc ~
$ ./Q2-4feb.sh
enter num1
1
enter num2
4
enter num3
7
7 is greatest
HP@neeyati-pc ~
$
```

Q3. Write a script to enter the marks of a student. If the marks are greater than 70 display grade A, if the grade is greater than 60 and less than 70 display grade B, else display "Fail".

Code:

```
E ~  
echo " enter marks of student"  
read marks  
if [ $marks -gt 70]  
then  
echo " A Grade "  
elif [ $marks -gt 60 ] && [ $marks -lt 70 ]  
then  
echo " B Grade "  
else  
echo " FAIL"  
fi  
~
```

Output:

```
HP@neeyati-pc ~  
$ ./Q3-4feb.sh  
enter marks of student  
65  
B Grade  
HP@neeyati-pc ~  
$
```

Q4. Write a script to calculate factorial of a number.

Code:

```
E ~  
echo " enter a number"  
read num  
factorial=1  
while [ $num -gt 1 ]  
do  
factorial=$((factorial * num))  
num=$((num-1))  
done  
echo $factorial  
~  
~
```

Output:

```
HP@neeyati-pc ~  
$ ./Q4-4feb.sh  
enter a number  
4  
24  
HP@neeyati-pc ~  
$ |
```

Q5. Write a script to display whether a user is valid or not.

Code:

```
#!/bin/bash
echo " enter name of user"
read username
if [ "$(whoami)" = "$username" ]
then
echo " valid user!"
else
echo " invalid user !"
fi
~
```

Output:

```
HP@neeyati-pc ~
$ ./Q5-4feb.sh
 enter name of user
HP
 valid user!

HP@neeyati-pc ~
$ ./Q5-4feb.sh
 enter name of user
jhuyhgiuk
 invalid user !

HP@neeyati-pc ~
$ |
```

Experiment -7

Objective: To write scripts for given questions

Software used: Cygwin64 Terminal

Q1. Write a script to reverse number passed using positional parameter.

Code:

```
#!/bin/bash
num=$1
rev=0
while [ $num -gt 0 ]
do
    rem=$(( $num % 10 ))
    rev=$(( $rev "x" 10 + $rem ))
    num=$(( $num / 10 ))
done
echo " the reverse of $1 is $rev "
```

Output:

```
HP@neeyati-pc ~
$ ./Lab7Q1.sh 78
the reverse of 78 is 87

HP@neeyati-pc ~
$ vi Lab7Q1.sh
```

Q2. Write a script to list all files in a directory using for loop.

Code:

```
#!/bin/bash
cd $1
for FILE in *;
do
    echo "$FILE"
done
```

Output:

```
$ chmod 777 Lab7Q2.sh

HP@neeyati-pc ~
$ ./Lab7Q2.sh
10febQ1.sh
10febQ3.sh
10febQ4.sh
10febQ5.sh
3febQ1.sh
Lab7Q1.sh
Lab7Q2.sh
Lab7Q6.sh
Program.sh
Program1.sh
Q2-21jan.sh
Q2-4feb.sh
Q3-4feb.sh
Q4-4feb.sh
Q5-4feb.sh
Question1.sh
Question2.sh
Question3.sh
addText
directorynew
hji
newname1
newname2
newname3
newnameos
newnameos1
```

Q3. Write a script to find largest value passed using command line

Code:

```
E ~
if [ "$#" = 0 ]
then
echo " No arguments have been passed"
exit 1
fi

largest=$1
for arg in "$@"
do

if [ "$arg" -gt "$largest" ]
then
largest=$arg
fi
done
echo " largest value passed is $largest"

~
```

Output:

```
HP@neeyati-pc ~
$ ./10febQ3.sh 7 9 5
largest value passed is 9

HP@neeyati-pc ~
$
```

Q4. Write a script to search particular file and rename it.

Code:

```
E ~
echo " enetr file name "
read filename
if [ "$(ls | grep $filename)" = "$filename" ]
then
echo "enter new name for file"
read Namenew
mv $filename $Namenew
else
echo " file not found"
fi
~
~
~
~
```

Output:

```
HP@neeyati-pc ~
$ ./10febQ3.sh
enetr file name
Q1-4feb.sh
enter new name for file
newname1

HP@neeyati-pc ~
```

Q5. Write a script to print fibonacci series.

Code:

```
#!/bin/bash
echo " enter the number of terms "
read terms
num1=0
num2=1
sum=0
for (( i=0; i< $terms; i++ ))
{
echo " $num1"
sum=$(( $num1 + $num2 ))
num1=$num2
num2=$sum
}
~
~
```

Output:

```
HP@neeyati-pc ~
$ ./10febQ5.sh
enter the number of terms
7
0
1
1
2
3
5
8
HP@neeyati-pc ~
$
```

Q6. Write a script to create and delete directory using case.

Code:

```
#!/bin/bash
echo " 1. make a directory "
echo " 2. delete a directory"
read choice
case $choice in
1)
echo " enter name for new directory"
read choice
mkdir $choice
;;
2)
echo " enter name of directory to be deleted"
read choice
rm -r $choice
;;
*)
echo " choose valid option "
;;
esac
~
~
~
~
```

Output:

```
HP@neeyati-pc ~
$ ./Lab7Q6.sh
 1. make a directory
 2. delete a directory
1
enter name for new directory
hjij

HP@neeyati-pc ~
$ ./Lab7Q6.sh
 1. make a directory
 2. delete a directory
2
enter name of directory to be deleted
new

HP@neeyati-pc ~
$
```

Experiment -8

Objective: To write a program to simulate FCFS CPU scheduling algorithm.
Verify the code with provided input.

Input1: P1,P2,P3; BT P1 3, P2 4 , P3 5 ; AT=0.

Input2:

Process Id	Arrival time	Burst time
P1	3	4
P2	5	3
P3	0	2
P4	5	1
P5	4	3

Code:

```
#include<iostream>
using namespace std;

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void SortAT(int P[], int n, int BT[], int AT[])
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
            if (AT[j] < AT[min_idx])
                min_idx = j;

        // Swap the found minimum element with the first element
        swap(&AT[min_idx], &AT[i]);
        swap(&P[min_idx], &P[i]);
        swap(&BT[min_idx], &BT[i]);
    }
    for (i = 0; i < n-1; i++)
    {
        for (j = i+1; j < n; j++)
        {
            if(AT[i] == AT[j])
```



```

        if(P[i]>P[j])
        {
            swap(&AT[i], &AT[j]);
            swap(&P[i], &P[j]);
            swap(&BT[i], &BT[j]); }
    }
}

```

```

void SortP(int P[], int n, int BT[], int AT[],int WT[],int TAT[])
{

```

```

    int i, j, min_idx;

```

```

    // One by one move boundary of unsorted subarray

```

```

    for (i = 0; i < n-1; i++)
    {

```

```

        // Find the minimum element in unsorted array

```

```

        min_idx = i;

```

```

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

```

```

        if (P[j] < P[min_idx])

```

```

            min_idx = j;

```

```

        // Swap the found minimum element with the first element

```

```

        swap(&P[min_idx], &P[i]);

```

```

        swap(&AT[min_idx], &AT[i]);

```

```

        swap(&BT[min_idx], &BT[i]);

```

```

        swap(&WT[min_idx], &WT[i]);

```

```

        swap(&TAT[min_idx], &TAT[i]);
    }
}

```

```

void Find_WT(int P[], int n, int BT[],int WT[], int AT[])
{

```

```

    // service time (ST) = summation of previous burst times

```

```

    int ST[n];

```

```

    // ST and WT is 0 for P1

```

```

    ST[0] = 0;

```

```

    WT[0] = 0;

```

```

    for (int i = 1; i < n ; i++)
    {

```

```

        ST[i] = ST[i-1] + BT[i-1];

```

```

    // WT for current process = ST - AT[i]

```

```

    if(AT[i] > ST[i])
    {

```

```

        ST[i]+=1;
    }

```

```

    WT[i] = ST[i] - AT[i];

```

```

// If WT for P is in -ve, P is in ready queue before CPU is idle, WT=0
    if (WT[i] < 0)
        WT[i] = 0;
    }
}

```

```

void Find_TAT(int P[], int n, int BT[], int WT[], int TAT[])
{
    for (int i = 0; i < n; i++)
        TAT[i] = BT[i] + WT[i];
}

```

```

void Find_AvgTime(int P[], int n, int BT[], int AT[])
{
    int WT[n], TAT[n];
    SortAT(P,n,BT,AT);
    Find_WT(P, n, BT,WT,AT);
    Find_TAT( P, n, BT, WT, TAT);
    //SortP( P, n, BT, AT, WT, TAT);
}

```

```

// Display processes along with all details
    cout << "P " << "\t BT\t" << "\tAT\t "
        << " \tWT \t" << "\t TAT"
        << "\tCT \n";
    int total_WT = 0, total_TAT = 0;
    for (int i = 0; i < n; i++)
    {
        total_WT = total_WT + WT[i];
        total_TAT = total_TAT + TAT[i];
        int CT = TAT[i] + AT[i];
        cout << P[i] << "\t " << BT[i] << "\t\t"
            << AT[i] << "\t\t" << WT[i] << "\t\t "
            << TAT[i] << "\t\t " << CT << endl;
    }

    cout << "Average waiting time = "
        << (float)total_WT / (float)n;
    cout << "\nAverage turn around time = "
        << (float)total_TAT / (float)n;
}

```

//with input1

```

int main()
{
    int P[] = {1, 2, 3};
    int n = sizeof P / sizeof P[0];
    int BT[] = {3,4,5};
    int AT[] = {0,0,0};
}

```

```

        Find_AvgTime( P , n, BT, AT);

    return 0;
}

```

Code:

//with input2

```

int main()
{
    int P[] = {1, 2, 3,4,5};
    int n = sizeof P / sizeof P[0];
    int BT[] = {4,3,2,1,3};
    int AT[] = {3,5,0,5,4};
    Find_AvgTime( P , n, BT, AT);

    return 0;
}

```

Output:

```

/tmp/082$wDegbw.o
P  BT   AT   WT   TAT  CT
1   3    0    0    3    3
2   4    0    3    7    7
3   5    0    7   12   12
Average waiting time = 3.33333
Average turn around time = 7.33333

```

```

/tmp/082$wDegbw.o
P  BT   AT   WT   TAT  CT
3   2    0    0    2    2
1   4    3    0    4    7
5   3    4    3    6   10
2   3    5    5    8   13
4   1    5    8    9   14
Average waiting time = 3.2
Average turn around time = 5.8

```

Experiment -9

Objective: To write a program to implement pre-emptive and non-pre-emptive SJF
Verify the code with provided input.

Input1:

<u>Process</u>	<u>Burst Time</u>
P_1	6
P_2	8
P_3	7
P_4	3

Input2:

<u>Proces</u>	<u>Arrival Time</u>	<u>Burst Time</u>
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Code:

//pre=emptive SJF

```
#include <stdio.h>
int main(){
    int AT[20],BT[20],temp[20];
    int i;
    int count=0;
    int smallest, t , limit ;
    double WT=0, TAT=0,end;
    float AvgWT,AvgTAT;

    printf("Enter the number of processes:");
    scanf("%d", &limit);
    printf("Details :\n");
    for(i=0;i<limit; i++)
    {
        printf("\nAT: ");
        scanf("%d",&AT[i]);
        printf("BT: ");
        scanf("%d",&BT[i]);
        printf("-----");
        temp[i]= BT[i];
    }
```

```

BT[9]=9999;
for (t=0 ; count!= limit;t++)
{
    smallest=9;
    for(i=0 ; i< limit;i++)
    {
        if (AT[i] < t && BT[i]<BT[smallest] && BT[i] >0)
        {smallest = i ;
        }
    }
    BT[smallest]--;
    if (BT[smallest]==0)
    {
        count++;
        end=t;
        WT=WT + end -AT[smallest] - temp[smallest];
        TAT = TAT + end-AT[smallest];
    }
}
printf("\nAVERAGE WAITING TIME : %lf \n",AvgWT= WT /limit );
printf("AVERAGE TURN AROUND TIME : %lf", AvgTAT = TAT /limit);
return 0;
}

```

```

Enter the number of processes:4
Details :

```

```

AT: 0
BT: 6
-----
AT: 0
BT: 8
-----
AT: 0
BT: 7
-----
AT: 0
BT: 3
-----

```

```

AVERAGE WAITING TIME : 7.000000
AVERAGE TURN AROUND TIME : 13.000000

```

```

...Program finished with exit code 0
Press ENTER to exit console.

```

```

Enter the number of processes:4
Details :

```

```

AT: 0
BT: 8
-----
AT: 1
BT: 4
-----
AT: 2
BT: 9
-----
AT: 3
BT: 5
-----

```

```

AVERAGE WAITING TIME : 6.500000
AVERAGE TURN AROUND TIME : 13.000000

```

```

...Program finished with exit code 0
Press ENTER to exit console.

```

Code:

//Non Pre-emptive SJF

```

#include <stdio.h>
int main(){
    int BT[20],WT[20],TAT[20] , P[20];
    int i , j ,n,temp, pos;

```

```

int total =0;
float AVGWT;
float AVGTAT;
printf("Enter number of processes : ");
scanf("%d",&n);
printf("Enter burst time of processes :\n");
for ( i = 0 ; i<n;i++)
{
    printf("P%d:" , i+1);
    scanf("%d" , &BT[i]);
    P[i]=i+1;
}

```

```

//sort BT

```

```

for ( i =0 ; i <n ; i++){
    pos =i ;
    for (j = i +1; j<n;j++){
        if(BT[j]<BT[pos])
            pos = j ;
    }
}

```

```

temp = BT[i];
BT[i] =BT[pos];
BT[pos]=temp;

```

```

temp =P[i];
P[i] = P[pos];
P[pos]=temp;
}

```

```

WT[0]=0;
for ( i =1;i<n;i++)
{
    WT[i]=0;
    for(j=0;j<i;j++)
        WT[i]=WT[i]+BT[j];
    total = total + WT[i];
}

```

```

AVGWT= (float)total/n;
total=0;
printf("\nP    BT    WT    TAT");
for ( i=0 ;i <n ; i++ )
{
    TAT[i]= BT[i]+WT[i];
    total = total + TAT[i];
}

```

```

    printf("\nP%d    %d    %d    %d", P[i],BT[i],WT[i],TAT[i]);
}

AVGTAT=(float) total/n;
printf("\n\nAVERAGE WAITING TIME = %f", AVGWT);
printf("\n\nAVERAGE TURN AROUND TIME = %f\n", AVGTAT);

}

```

```

Enter number of processes : 4
Enter burst time of processes :
P1:6
P2:8
P3:7
P4:3

P      BT      WT      TAT
P4      3      0      3
P1      6      3      9
P3      7      9     16
P2      8     16     24

AVERAGE WAITING TIME = 7.000000
AVERAGE TURN AROUND TIME = 13.000000

...Program finished with exit code 0
Press ENTER to exit console.

```

```

Enter number of processes : 4
Enter burst time of processes :
P1:8
P2:4
P3:9
P4:5

P      BT      WT      TAT
P2      4      0      4
P4      5      4      9
P1      8      9     17
P3      9     17     26

AVERAGE WAITING TIME = 7.500000
AVERAGE TURN AROUND TIME = 14.000000

...Program finished with exit code 0
Press ENTER to exit console.

```

Experiment 10

Objective: To write a program to implement Round Robin Scheduling Algorithm.
Verify the code with provided input.

Q 7. Consider a set of 5 processes whose arrival time, CPU Times needed are given below.

Process	CPU Time	Arrival Time	priority
P1	10	0	5
P2	5	0	2
P3	3	2	1
P4	20	5	4
P4	2	10	3

Calculate Average Waiting Time and Turn Around Time for pre-emptive priority and Round Robin(Time Quantum= 4)

Code:

```
#include<stdio.h>
int main()
{
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
    float average_wait_time, average_turnaround_time;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    x = limit;
    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Details of Process[%d]\n", i + 1);
        printf("Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }
    printf("\nEnter Time Quantum:\t");
    scanf("%d", &time_quantum);
    printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");
    for(total = 0, i = 0; x != 0;)
    {
        if(temp[i] <= time_quantum && temp[i] > 0)
        {
            total = total + temp[i];
            temp[i] = 0;
            counter = 1;
        }
        else if(temp[i] > 0)
        {
```



```

        temp[i] = temp[i] - time_quantum;
        total = total + time_quantum;
    }
    if(temp[i] == 0 && counter == 1)
    {
        x--;
        printf("\nProcess[%d]\t\t%d\t\t %d\t\t %d", i + 1, burst_time[i], total -
arrival_time[i], total - arrival_time[i] - burst_time[i]);
        wait_time = wait_time + total - arrival_time[i] - burst_time[i];
        turnaround_time = turnaround_time + total - arrival_time[i];
        counter = 0;
    }
    if(i == limit - 1)
    {
        i = 0;
    }
    else if(arrival_time[i + 1] <= total)
    {
        i++;
    }
    else
    {
        i = 0;
    }
}
average_wait_time = wait_time * 1.0 / limit;
average_turnaround_time = turnaround_time * 1.0 / limit;
printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
printf("\n\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
return 0;
}

```

Output:

```
Enter Total Number of Processes:      5

Enter Details of Process[1]
Arrival Time:    0
Burst Time:      10

Enter Details of Process[2]
Arrival Time:    0
Burst Time:      5

Enter Details of Process[3]
Arrival Time:    2
Burst Time:      3

Enter Details of Process[4]
Arrival Time:    5
Burst Time:      20

Enter Details of Process[5]
Arrival Time:    10
Burst Time:      2

Enter Time Quantum:      4

Process ID    Burst Time    Turnaround Time    Waiting Time
Process[3]    3            9                6
Process[5]    2            7                5
Process[2]    5            22               17
Process[1]    10           28               18
Process[4]    20           35               15

Average Waiting Time:  12.200000
Avg Turnaround Time:  20.200001

...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment -11

Objective: To WAP to implement Banker's Algorithm. Verify the code with provided input.

Code:

```
#include<stdio.h>
#include<stdlib.h>
void print(int x[][10],int n,int m){
int i,j;
for(i=0;i<n;i++){
printf("\n");
for(j=0;j<m;j++){
printf("%d\t",x[i][j]); }
}
}
void res_request(int A[10][10],int N[10][10],int AV[10][10],int pid,int m)
{
int reqmat[1][10];
int i;
printf("\n Enter additional request :- \n");
for(i=0;i<m;i++){
printf(" Request for resource %d : ",i+1);
scanf("%d",&reqmat[0][i]);
}
for(i=0;i<m;i++)
if(reqmat[0][i] > N[pid][i]){
printf("\n Error encountered.\n");
exit(0);
}
for(i=0;i<m;i++)
if(reqmat[0][i] > AV[0][i]){
printf("\n Resources unavailable.\n");
exit(0);
}
for(i=0;i<m;i++){
AV[0][i]-=reqmat[0][i];
A[pid][i]+=reqmat[0][i];
N[pid][i]-=reqmat[0][i];
}
}
int safety(int A[][10],int N[][10],int AV[1][10],int n,int m,int a[]){
int i,j,k,x=0;
int F[10],W[1][10];
int pflag=0,flag=0;
for(i=0;i<n;i++)
F[i]=0;
for(i=0;i<m;i++)
```

```

W[0][i]=AV[0][i];
for(k=0;k<n;k++){
for(i=0;i<n;i++){
if(F[i] == 0){
flag=0;
for(j=0;j<m;j++){
if(N[i][j] > W[0][j])
flag=1;
}
if(flag == 0 && F[i] == 0){
for(j=0;j<m;j++)
W[0][j]+=A[i][j];
F[i]=1;
pflag++;
a[x++]=i; }
}
}
if(pflag == n)
return 1;
}
return 0;
}

void accept(int A[][10],int N[][10],int M[10][10],int W[1][10],int *n,int *m){
int i,j;
printf("\n Enter total no. of processes : ");
scanf("%d",n);
printf("\n Enter total no. of resources : ");
scanf("%d",m);
for(i=0;i<*n;i++){
printf("\n Process %d\n",i+1);
for(j=0;j<*m;j++){
printf(" Allocation for resource %d : ",j+1);
scanf("%d",&A[i][j]);
printf(" Maximum for resource %d : ",j+1);
scanf("%d",&M[i][j]);
}
}
printf("\n Available resources : \n");
for(i=0;i<*m;i++){
printf(" Resource %d : ",i+1);
scanf("%d",&W[0][i]);
}
for(i=0;i<*n;i++)
for(j=0;j<*m;j++)
N[i][j]=M[i][j]-A[i][j];
printf("\n Allocation Matrix");
print(A,*n,*m);
printf("\n Maximum Requirement Matrix");
print(M,*n,*m);
printf("\n Need Matrix");

```

```

print(N,*n,*m);
}
int banker(int A[][10],int N[][10],int W[1][10],int n,int m){
int j,i,a[10];
j=safety(A,N,W,n,m,a);
if(j != 0 ){
printf("\n\n");
for(i=0;i<n;i++)
printf(" P%d ",a[i]);
printf("\n A safety sequence has been detected.\n");
return 1;
}else{
printf("\n Deadlock has occured.\n");
return 0;
}
}
int main(){
int ret;
int A[10][10];
int M[10][10];
int N[10][10];
int W[1][10];
int n,m,pid,ch;
printf("\n DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n");
accept(A,N,M,W,&n,&m);
ret=banker(A,N,W,n,m);
if(ret !=0 ){
printf("\n Do you want make an additional request ? (1=Yes|0=No)");
scanf("%d",&ch);
if(ch == 1){
printf("\n Enter process no. : ");
scanf("%d",&pid);
res_request(A,N,W,pid-1,m);
ret=banker(A,N,W,n,m);
if(ret == 0 )
exit(0);
}
}else
exit(0);
return 0;
}

```

Consider a system that contains five processes P1, P2, P3, P4, P5 and the three resource types A, B and C. Following are the resources types: A has 10, B has 5 and the resource type C has 7 instances.

Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2
P2	2	0	0	3	2	2			
P3	3	0	2	9	0	2			
P4	2	1	1	2	2	2			
P5	0	0	2	4	3	3			

Output:

```

Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Maximum Requirement Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Need Matrix
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1

P1 P3 P4 P0 P2
A safety sequence has been detected.

Do you want make an additional request ? (1=Yes|0=No)1

Enter process no. : 1

Enter additional request :-
Request for resource 1 : 1
Request for resource 2 : 0
Request for resource 3 : 0

P1 P3 P4 P0 P2
A safety sequence has been detected.

```

DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM

Enter total no. of processes : 5

Enter total no. of resources : 3

```

Process 1
Allocation for resource 1 : 0
Maximum for resource 1 : 7
Allocation for resource 2 : 1
Maximum for resource 2 : 5
Allocation for resource 3 : 0
Maximum for resource 3 : 3

```

```

Process 2
Allocation for resource 1 : 2
Maximum for resource 1 : 3
Allocation for resource 2 : 0
Maximum for resource 2 : 2
Allocation for resource 3 : 0
Maximum for resource 3 : 2

```

```

Process 3
Allocation for resource 1 : 3
Maximum for resource 1 : 9
Allocation for resource 2 : 0
Maximum for resource 2 : 0
Allocation for resource 3 : 2
Maximum for resource 3 : 2

```

```

Process 4
Allocation for resource 1 : 2
Maximum for resource 1 : 2
Allocation for resource 2 : 1
Maximum for resource 2 : 2
Allocation for resource 3 : 1
Maximum for resource 3 : 2

```

Experiment -12

Objective: To write a program to implement LOOK Disc Scheduling Algorithm.

Code:

```
int size = 8;
#include <bits/stdc++.h>
using namespace std;

int diskSize = 200;

void LOOK(int a[], int head, string D)
{
    int seekCount = 0;
    int dist, currentTrack;
    vector<int> L, R;
    vector<int> seekSequence;

    for (int i = 0; i < size; i++) {
        if (a[i] < head)
            L.push_back(a[i]);
        if (a[i] > head)
            R.push_back(a[i]);
    }

    // sorting left and right vectors

    std::sort(L.begin(), L.end());
    std::sort(R.begin(), R.end());

    // run the while loop to scan right and to scan left , so two times.
    int run = 2;
    while (run--) {
        if (D == "left") {
            for (int i = L.size() - 1; i >= 0; i--) {
                currentTrack = L[i];

                // appending current track to seek sequence
                seekSequence.push_back(currentTrack);

                // calculate absolute distance
                dist = abs(currentTrack - head);

                // increase the total count
                seekCount += dist;

                // accessed track is now the new head
            }
        }
    }
}
```

```

        head = currentTrack;
    }
    // reversing the direction
    D = "right";
}
else if (D=="right") {
    for (int i = 0; i < R.size(); i++) {
        currentTrack = R[i];
        // appending current track to seek sequence
        seekSequence.push_back(currentTrack);

        // calculate absolute distance
        dist= abs(currentTrack - head);

        // increase the total count
        seekCount += dist ;

        // accessed track is now new head
        head = currentTrack;
    }
    // reversing the direction
    D = "left";
}

cout << "Total number of seek operations = "
    << seekCount << endl;

cout << "Seek Sequence is" << endl;

for (int i = 0; i < seekSequence.size(); i++) {
    cout << seekSequence[i] << endl;
}

}

int main()
{
    int a[size] = { 150 , 89, 25, 40 , 97, 22, 59 , 107 };
    int head = 40;
    string D= "right";

    cout << "Initial position of head: "
        << head << endl;

    LOOK(a, head, D);

    return 0;
}

```


Output:

```
Initial position of head: 40
Total number of seek operations = 238
Seek Sequence is
59
89
97
107
150
25
22

...Program finished with exit code 0
Press ENTER to exit console. 
```