SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

DEPARTMENT OF INFORMATION TECHNOLOGY

II B.Tech- Information Technology

22CS403 - OPERATING SYSTEMS LAB

PRACTICAL RECORD

Submitted by

Name : ………………………

Reg.No : ………………………

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

DEPARTMENT OF INFORMATION TECHNOLOGY

22CS403 - OPERATING SYSTEMS LAB

PRACTICAL RECORD

Name : ………………………… **Reg.no :……………..…………**

Class : III BTECH IT C **Semester : IV**

BONAFIDE CERTIFICATE

Certified bonafide record **of work done by Mr. /Ms …………………………. Reg No. …………………….….. during the academic year 2023-2024 Submitted for the end semester practical examination held on ………......**

Staff-In Charge **HOD**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

Department of IT

Rubrics for Evaluating Laboratory

Subject Code : 22CS403

Lab Name : Operating Systems Lab

Method: Lab Reports and Observation of Faculty Incharge

Outcomes Assessed:

a) Graduates will demonstrate knowledge of mathematical, scientific and multidisciplinary approach for problem solving.

b) Graduates will be able to apply their knowledge in various programming skills to create solutions for product based and application based software.

c) Graduates will possess the ability to create real time solutions for different projects by using modern tools prevailing in the current trends.

e) Graduates attain advanced knowledge in the stream of Information Technology and basic knowledge in Electronics and Communication Engineering to develop and maintain the simple and complex information systems.

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

Department of IT

Register Number :

Name of the Student :

Name of the lab : 22CS403 Operating Systems Lab

Components	**Exp No and Date**												**Average Score**
	Ex1	**Ex2**	**Ex3**	**Ex4**	**Ex5**	**Ex6**	**Ex7**	**Ex8**	**Ex9**	**Ex10**	**Ex11**	**Ex12**	
Aim & Algorithm 20 Marks													
Coding 30 Marks													
Compilation & Debugging 30 Marks													
Execution & Results 10 Marks													
Documentation & Viva 10 Marks													
Total													

Staff In-charge

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

PROGRAMME OUTCOMES

a) Graduates will demonstrate knowledge of mathematical, scientific and multidisciplinary approach for problem solving. ***(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging)***

b) Graduates will be able to apply their knowledge in various programming skills to create solutions for product based and application based software. ***(Criteria to be used for assessment Coding, Compilation and Debugging)***

c) Graduates will possess the ability to create real time solutions for different projects by using modern tools prevailing in the current trends. ***(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging, Execution and Results (Inclusion of Generalization like Subroutines, Modules)***

d) Graduates attain advanced knowledge in the stream of Information Technology and basic knowledge in Electronics and Communication Engineering to develop and maintain the simple and complex information systems. ***(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging, Execution and Results (Inclusion of Generalization like Subroutines, Modules***

Staff In-charge

Ex No : 1	Basic Linux Commands
03.05.2024	

AIM

To study the basic commands in Linux.

COMMANDS

1. **TASK** **:** To display user defined message

   Command **:** echo

   Syntax **:** echo "Message"

   Explanation : This command displays the message after echo command on the screen.

2. **TASK** **:** To create a file

   Command **:** vi

   Syntax **:** vi filename

   Explanation : This command creates the file and content can be typed.

3. **TASK** **:** To view a file

   Command **:** cat

   Syntax **:** cat filename

   Explanation : This command displays the contents of specified file.

4. **TASK** **:** To display the files and folders present in the Login

   Command **:** ls

   Syntax **:** ls

   Explanation : This command displays the files and folders present in login.

5. **TASK** **:** To copy a file

   Command **:** cp

   Syntax **:** cp sourcefile destfile

   Explanation : This command produces a copy of the stored file and is stored

6. **TASK** **:** To rename or move a file

   Command **:** mv

   Syntax **:** mv sourcefile destfile

   Explanation : Command moves content of source file to destination, then source file is deleted

7. **TASK** **:** To display the no. of characters in a file

   Command **:** wc

   Syntax **:** wc filename

   Explanation : The command displays the no. of lines, words and characters of file

8. **TASK** **:** To display the online manual

   Command **:** man

   Syntax **:** man ls

   Explanation : This command displays information about ls

9. **TASK** **:** To retrieve a part of the file

   Command **:** head

   Syntax **:** head –noofrows filename

   Explanation : This command displays no. of rows from the top of the specified file.

10. **TASK** **:** To retrieve a part of the file

    Command **:** tail

    Syntax **:** tail –noofrows filename

    Explanation : This command displays no. of rows from the bottom of the specified file.

11. **TASK** **:** To change a directory

    Command **:** cd

    Syntax **:** cd dirname

    Explanation : This command switch one directory to another.

12. **TASK** **:** To create a directory

    Command **:** mkdir

    Syntax **:** mkdir dirname

    Explanation : This command creates a new directory with specified name.

13. **TASK** **:** To delete a file

    Command **:** rm

    Syntax **:** rm filename

    Explanation : This command deletes the specified file from directory

14. **TASK** **:** To delete a directory

    Command **:** rmdir

    Syntax **:** rmdir dirname

    Explanation : This command deletes the specified directory

15. **TASK** **:** To sort the contents of a file

    Command **:** sort

    Syntax **:** sort filename

    Explanation : This command sorts the contents of a file in ascending order.

16. **TASK** **:** To compress a given file or directory
Command **:** gzip

Syntax **:** gzip filename

Explanation : This command compress a given file or directory

17. **TASK** **:** To compare the contents of two files

Command **:** cmp

Syntax **:** cmp f1 f2

Explanation : The command compares a given file and displays the area which it differs

18. **TASK** **:** To find the difference between the contents of two files

Command **:** diff

Syntax **:** diff f1 f2

Explanation : The command compares a given file and displays the area from which it differs

19. **TASK** **:** To display the current working directory

Command **:** pwd

Syntax **:** pwd

Explanation : This displays current working directory showing this path.

20. **TASK** **:** To display calendar of current month

Command **:** cal

Syntax **:** cal 2016

Explanation : This displays the calendar of the current month on screen

21. **TASK** **:** To come out of sub directory

Command **:** cd

Syntax **:** cd ..

Explanation : This command helps is switch to main directory.
22. **TASK** **:** To display the user details
Command **:** whoami
Syntax **:** whoami
Explanation : This command displays current user of the system on the screen.
23. **TASK** **:** To match given pattern
Command **:** grep
Syntax **:** grep [- v –n -c] pattern filename
Explanation : command verifies filename and checks whether pattern present in file or not.

24. **TASK** **:** To gives permissions to given file

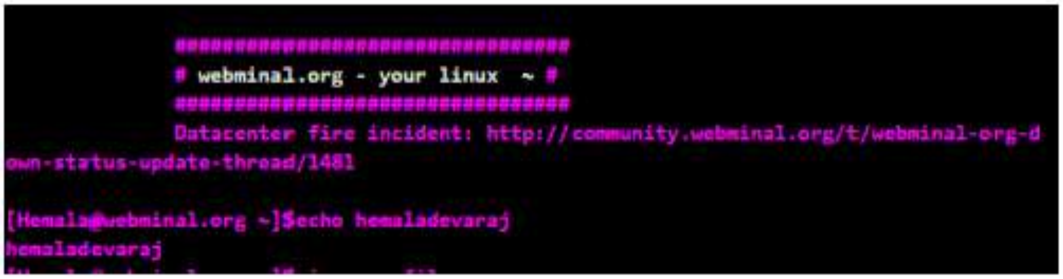
Command **:** chmod

Syntax **:** chmod 777 filename

Explanation : This command gives the corresponding permission to user, group and other.

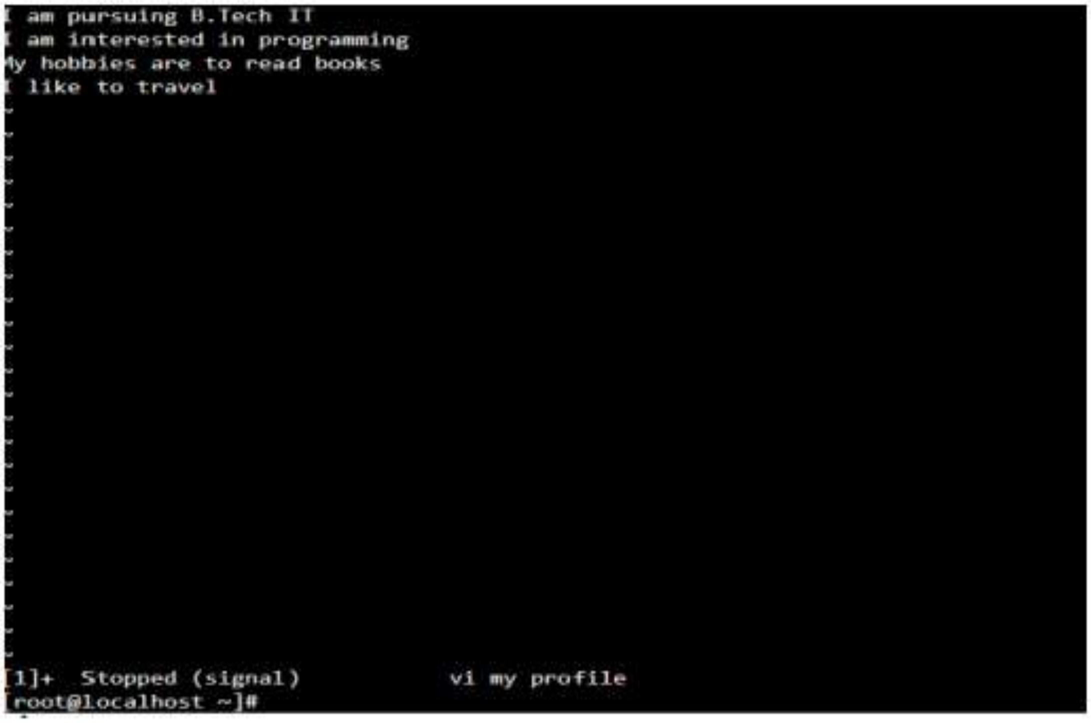
OUTPUT:

1)ECHO:



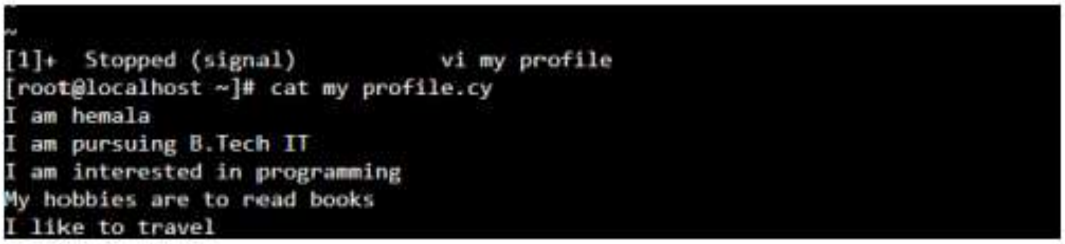
```
            ####################################
            # webminal.org - your linux  ~ #
            ####################################
            Datacenter fire incident: http://community.webminal.org/t/webminal-org-d
own-status-update-thread/1481

[Hemala@webminal.org ~]$echo hemaladevaraj
hemaladevaraj
```

2)VI:



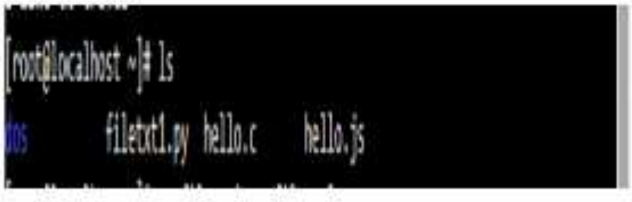
```
I am pursuing B.Tech IT
I am interested in programming
My hobbies are to read books
I like to travel
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
[1]+  Stopped (signal)           vi my profile
[root@localhost ~]#
```

3)CAT:



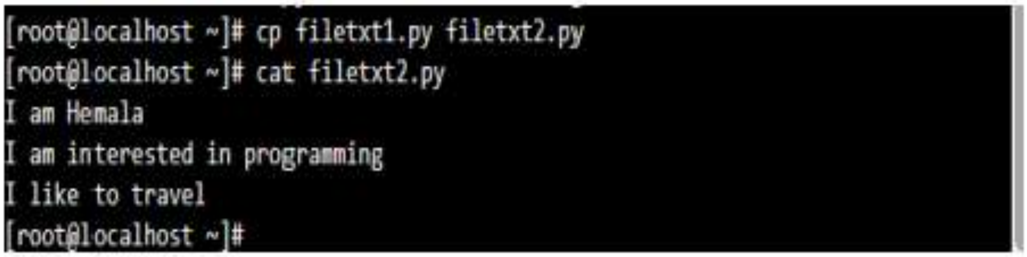
```
~
[1]+  Stopped (signal)           vi my profile
[root@localhost ~]# cat my profile.cy
I am hemala
I am pursuing B.Tech IT
I am interested in programming
My hobbies are to read books
I like to travel
```

4)ls:



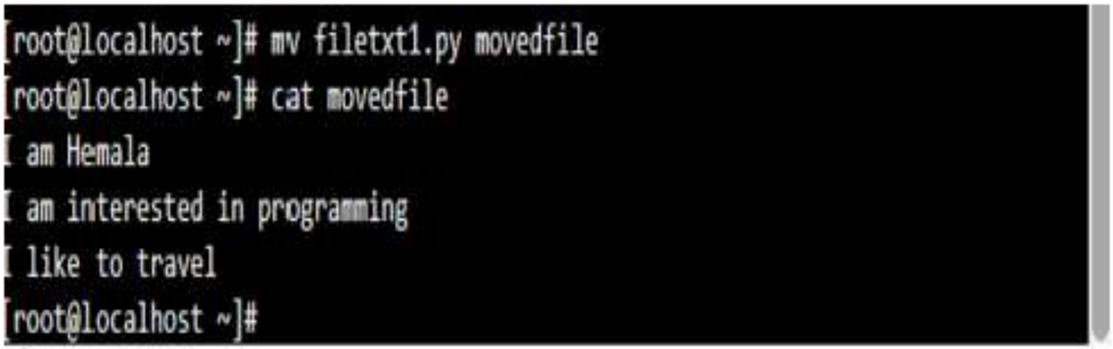
```
[root@localhost ~]# ls
dos        filetxt1.py hello.c    hello.js
```

5)cp:



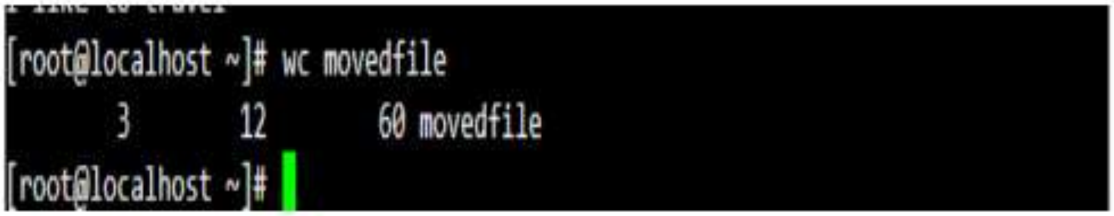
```
[root@localhost ~]# cp filetxt1.py filetxt2.py
[root@localhost ~]# cat filetxt2.py
I am Hemala
I am interested in programming
I like to travel
[root@localhost ~]#
```

6) MV:



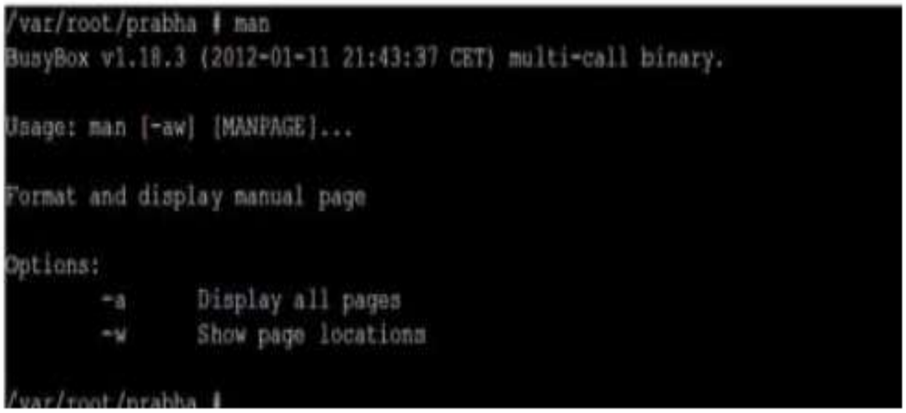
```
[root@localhost ~]# mv filetxt1.py movedfile
[root@localhost ~]# cat movedfile
I am Hemala
I am interested in programming
I like to travel
[root@localhost ~]#
```

7)wc:



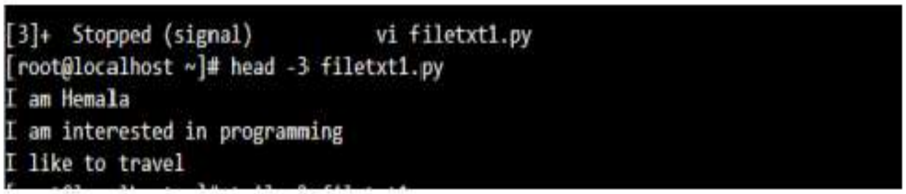
```
[root@localhost ~]# wc movedfile
      3      12       60 movedfile
[root@localhost ~]#
```

8)MAN:



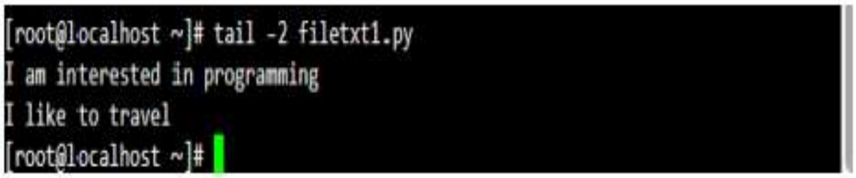
```
/var/root/prabha # man
BusyBox v1.18.3 (2012-01-11 21:43:37 CET) multi-call binary.

Usage: man [-aw] [MANPAGE]...

Format and display manual page

Options:
        -a      Display all pages
        -w      Show page locations

/var/root/prabha #
```

9)HEAD:



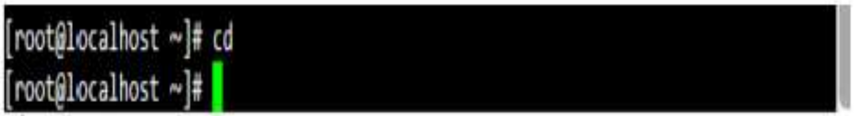
```
[3]+  Stopped (signal)          vi filetxt1.py
[root@localhost ~]# head -3 filetxt1.py
I am Hemala
I am interested in programming
I like to travel
[root@localhost ~]# tail -2 filetxt1
```

10)TAIL:



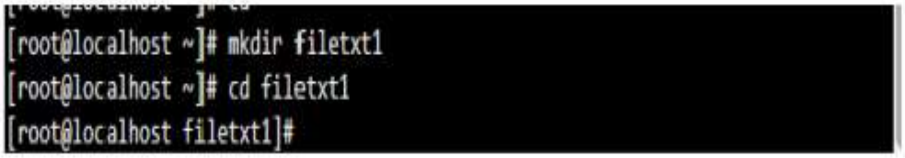
```
[root@localhost ~]# tail -2 filetxt1.py
I am interested in programming
I like to travel
[root@localhost ~]#
```

11)CD:



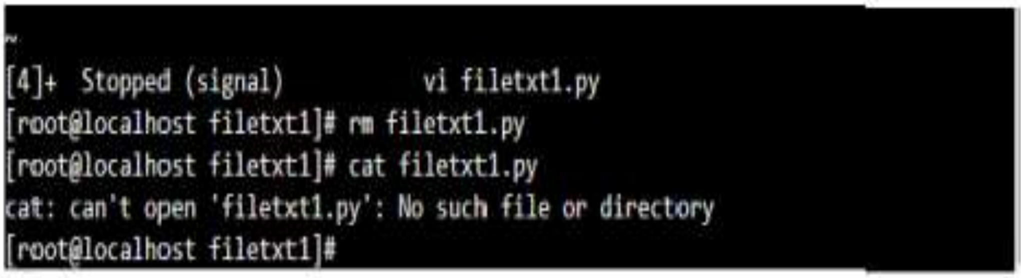
```
[root@localhost ~]# cd
[root@localhost ~]#
```

12)MKDIR:



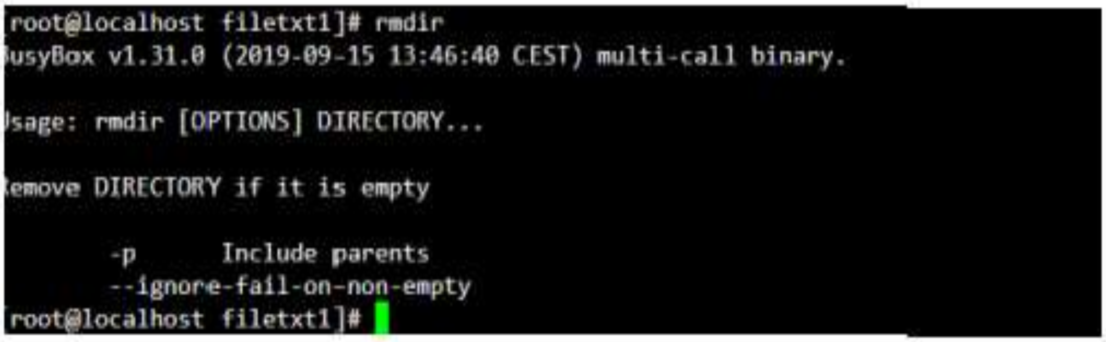
```
[root@localhost ~]# cd
[root@localhost ~]# mkdir filetxt1
[root@localhost ~]# cd filetxt1
[root@localhost filetxt1]#
```

13)RM:



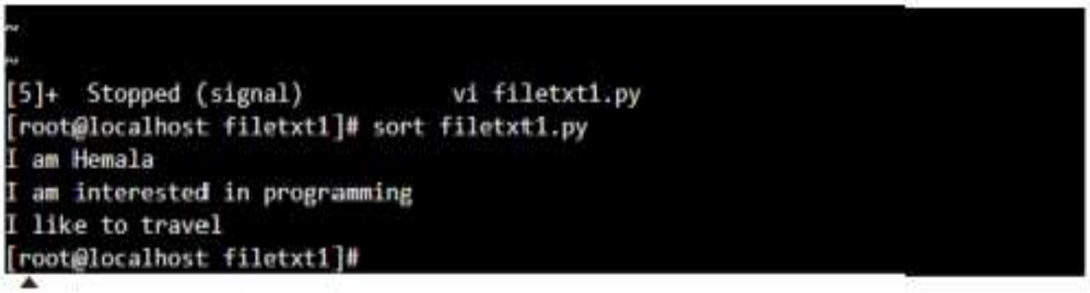
```
~
[4]+  Stopped (signal)          vi filetxt1.py
[root@localhost filetxt1]# rm filetxt1.py
[root@localhost filetxt1]# cat filetxt1.py
cat: can't open 'filetxt1.py': No such file or directory
[root@localhost filetxt1]#
```

14)RMDIR:



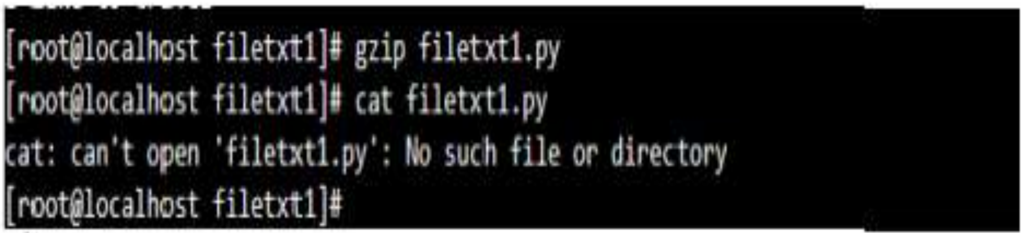
```
root@localhost filetxt1]# rmdir
usyBox v1.31.0 (2019-09-15 13:46:40 CEST) multi-call binary.

sage: rmdir [OPTIONS] DIRECTORY...

emove DIRECTORY if it is empty

        -p      Include parents
        --ignore-fail-on-non-empty
root@localhost filetxt1]#
```

15)SORT:



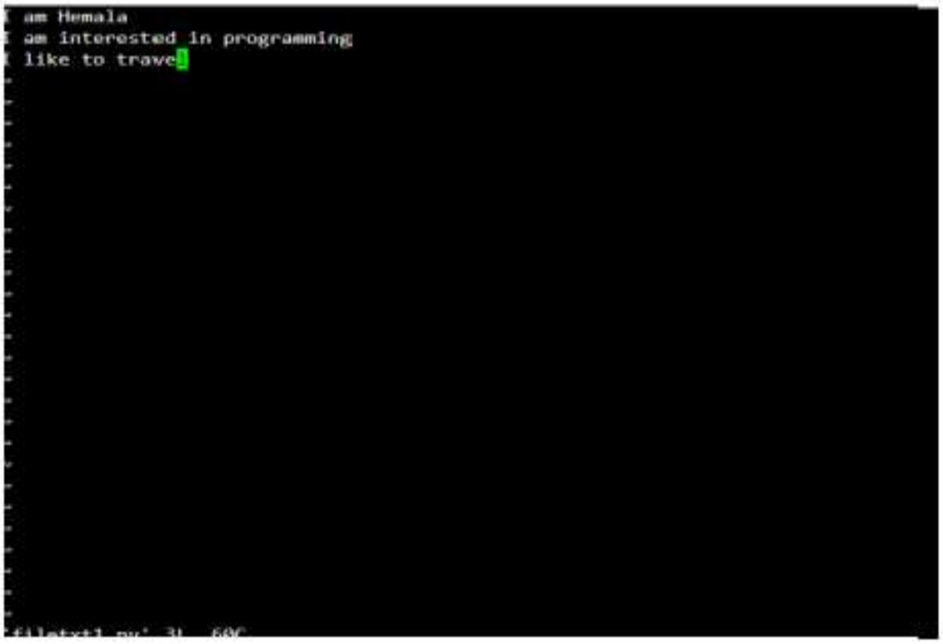
```
~
~
[5]+  Stopped (signal)          vi filetxt1.py
[root@localhost filetxt1]# sort filetxt1.py
I am Hemala
I am interested in programming
I like to travel
[root@localhost filetxt1]#
```

16)GZIP:

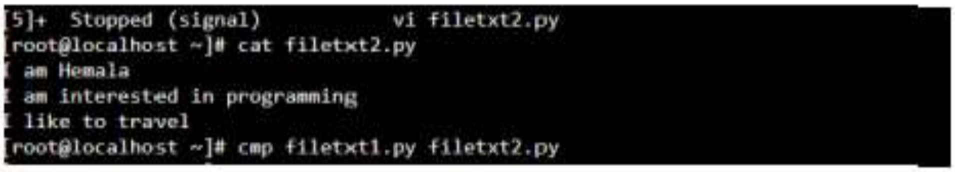


```
[root@localhost filetxt1]# gzip filetxt1.py
[root@localhost filetxt1]# cat filetxt1.py
cat: can't open 'filetxt1.py': No such file or directory
[root@localhost filetxt1]#
```

17)CMP:

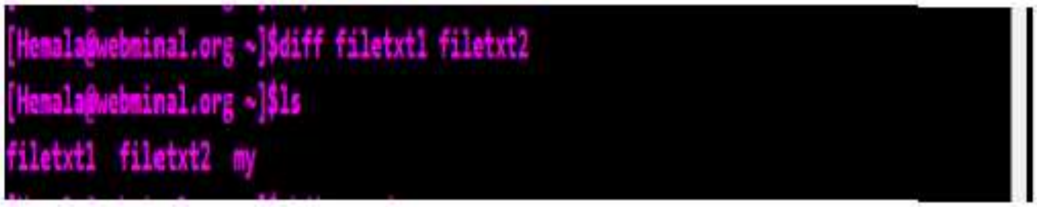


```
I am Hemala
I am interested in programming
I like to travel
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"filetxt1.py" 3L, 60C
```



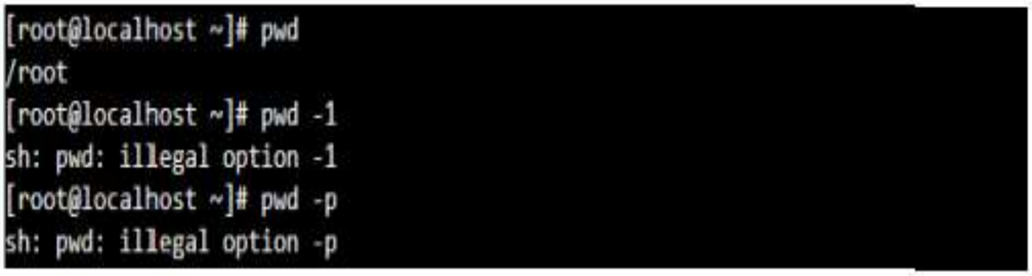
```
[5]+  Stopped (signal)           vi filetxt2.py
[root@localhost ~]# cat filetxt2.py
I am Hemala
I am interested in programming
I like to travel
[root@localhost ~]# cmp filetxt1.py filetxt2.py
```

18)DIFF:



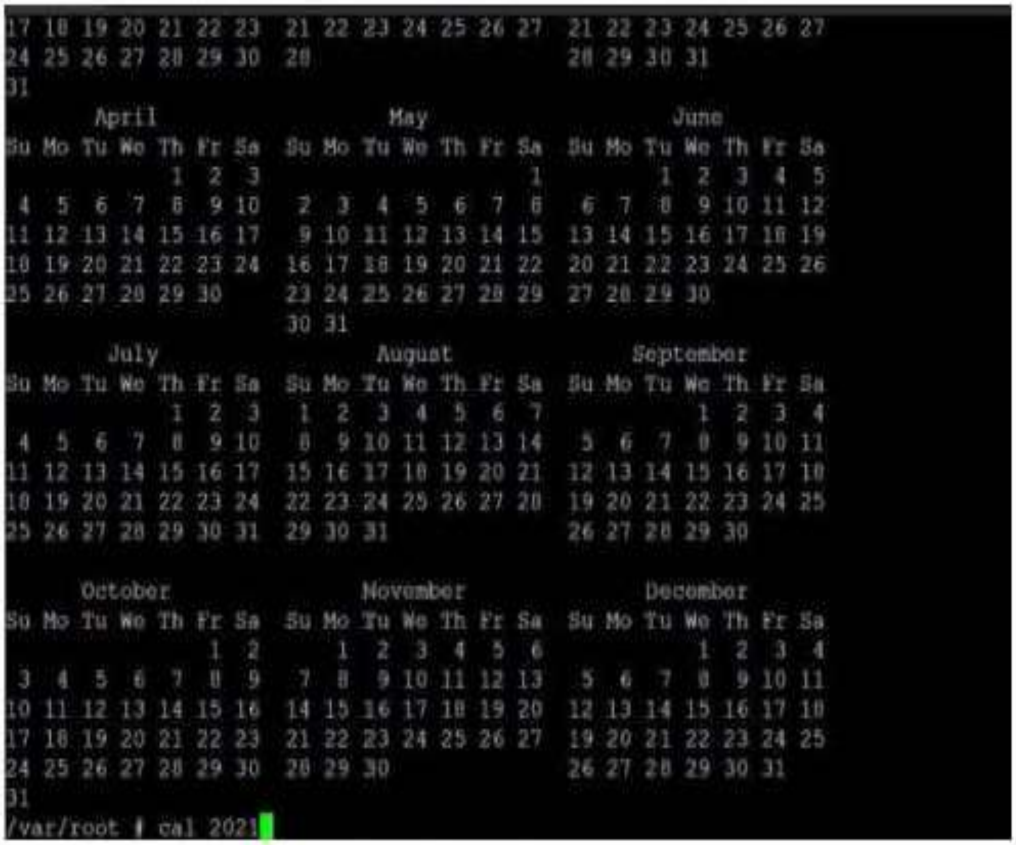
```
[Hemala@webminal.org ~]$diff filetxt1 filetxt2
[Hemala@webminal.org ~]$ls
filetxt1  filetxt2  my
```

19)PWD:



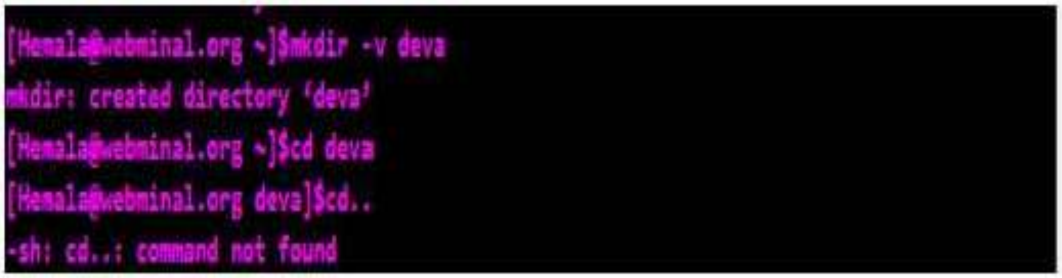
```
[root@localhost ~]# pwd
/root
[root@localhost ~]# pwd -1
sh: pwd: illegal option -1
[root@localhost ~]# pwd -p
sh: pwd: illegal option -p
```

20)CAL:



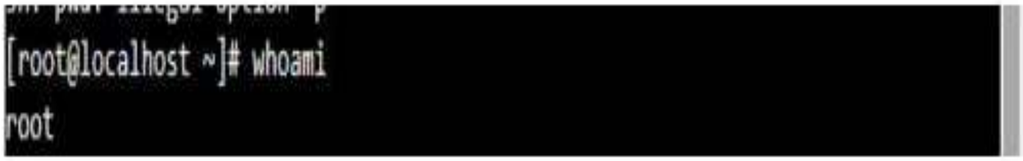
```
17 18 19 20 21 22 23  21 22 23 24 25 26 27  21 22 23 24 25 26 27
24 25 26 27 28 29 30  28                    28 29 30 31
31
       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3                     1         1  2  3  4  5
 4  5  6  7  8  9 10   2  3  4  5  6  7  8   6  7  8  9 10 11 12
11 12 13 14 15 16 17   9 10 11 12 13 14 15  13 14 15 16 17 18 19
18 19 20 21 22 23 24  16 17 18 19 20 21 22  20 21 22 23 24 25 26
25 26 27 28 29 30     23 24 25 26 27 28 29  27 28 29 30
                      30 31
        July                 August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7            1  2  3  4
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   5  6  7  8  9 10 11
11 12 13 14 15 16 17  15 16 17 18 19 20 21  12 13 14 15 16 17 18
18 19 20 21 22 23 24  22 23 24 25 26 27 28  19 20 21 22 23 24 25
25 26 27 28 29 30 31  29 30 31              26 27 28 29 30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6            1  2  3  4
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   5  6  7  8  9 10 11
10 11 12 13 14 15 16  14 15 16 17 18 19 20  12 13 14 15 16 17 18
17 18 19 20 21 22 23  21 22 23 24 25 26 27  19 20 21 22 23 24 25
24 25 26 27 28 29 30  28 29 30              26 27 28 29 30 31
31
/var/root # cal 2021
```

21)CD:



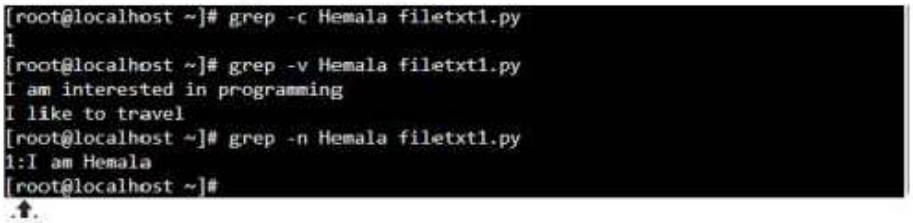
```
[Hemala@webminal.org ~]$mkdir -v deva
mkdir: created directory ‘deva’
[Hemala@webminal.org ~]$cd deva
[Hemala@webminal.org deva]$cd..
-sh: cd..: command not found
```

22)WHOMAI:



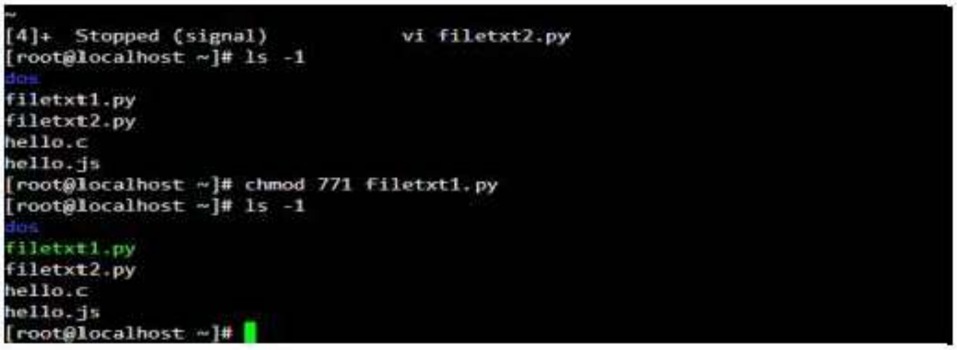
```
sh: pwd: illegal option -p
[root@localhost ~]# whoami
root
```

23)GREP:



```
[root@localhost ~]# grep -c Hemala filetxt1.py
1
[root@localhost ~]# grep -v Hemala filetxt1.py
I am interested in programming
I like to travel
[root@localhost ~]# grep -n Hemala filetxt1.py
1:I am Hemala
[root@localhost ~]#
```

24)CHMOD:



```
~
[4]+  Stopped (signal)          vi filetxt2.py
[root@localhost ~]# ls -1
dos
filetxt1.py
filetxt2.py
hello.c
hello.js
[root@localhost ~]# chmod 771 filetxt1.py
[root@localhost ~]# ls -1
dos
filetxt1.py
filetxt2.py
hello.c
hello.js
[root@localhost ~]#
```

RESULT:

The above program has been executed and the output has been shown successfully.

Ex No : 2	Use of control structures in Shell Programming
06.05.2024	

AIM

To evaluate the use of shell programming control Structures.

DESCRIPTION

(a) if.....fi structure:

The if... fi structure allows shell to make decisions and execute statements conditionally.

```
SYNTAX:   if [expression]
            then statements
            fi
```

(b) if....... else statement:

If else statements can be used to select an option from a given set of options

```
SYNTAX: if [expression]
          then
                 Statement(s)
          else
                Statement
          fi
```

(c) while loop:

The while loop enables us to execute a set of commands repeatedly until some condition occurs

```
SYNTAX: while [condition]
          do
          Statement (5)
          done
```

(d) **for loop**

The for loop operates on list of items it repeals a set of commands for every item in a list.

```
SYNTAX: for((int i=o; i<n; i++)
do
      Statement(s)
done.
```

Programs

a) To find whether a given voter is eligible to vote.

AIM:

To find whether a given voter is eligible to vote

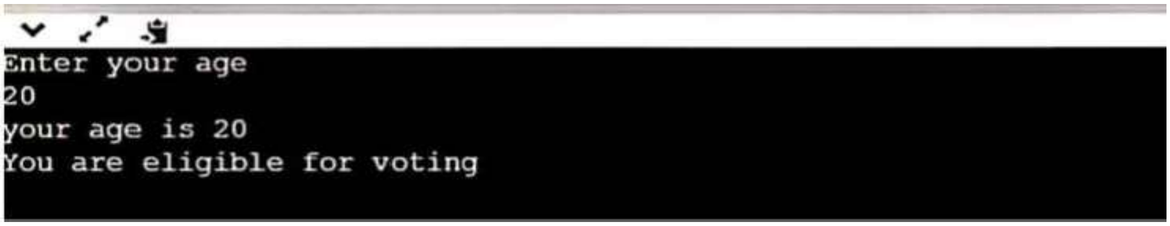
ALGORITHM:

1. Start
2. Declare a variable to enter age
3. And check whether the age is above or equal to 18
4. If it is above 18, thin the voter is eligible to vote.
5. If the age is below 18 then the rota is not eligible to vote.
6. End

PROGRAM:

```
echo Enter your age
read age
echo your age is $age
if [ $age -ge 18]
then
echo You are eligible for voting
else
echo You are not eligible for voting
fi
```

OUTPUT:



```
Enter your age
20
your age is 20
You are eligible for voting
```

b) To find the grade of a student using multiple if

AIM:

To find the grade of a student using multiple if

ALGORITHM:

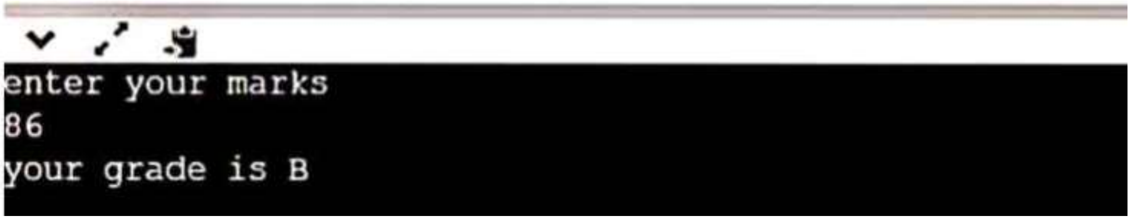
1. Start
2. Declare a variable to get mark a student
3. If mark> 90 then grade A, if mark >= 80 then grade B.
4. If mark>=70 then grade C, if mark>=60 then grade b
5. If mark>= 50 then grade E else grade F.
6. End.

PROGRAM:

```
echo enter your marks
read mark
if [ $mark -ge 90 ]
then
echo your grade is A
elif [ $mark -ge 80 ]
then
echo your grade is B
elif [ $mark -ge 70 ]
then
echo your grade is C
elif [ $mark -ge 60 ]
then
```

```
echo your grade is D
elif [ $mark -ge 50 ]
then
echo your grade is E
else
echo you failed
fi
```

OUTPUT:



```
enter your marks
86
your grade is B
```

c) To find whether the given year is a leap year or not

AIM:

To find whether the given year is a leap year or not

ALGORITHM:

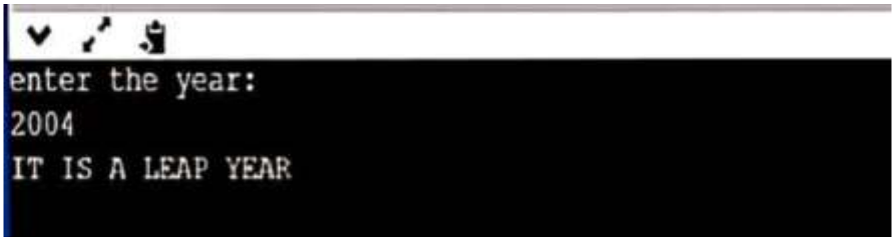
1. Start
2. Declare a variable to enter year.
3. Divide the year by 4/ remainder
4. If the remainder is equal to o then its leap year.
5. If its not zero then its not a leap year
6. End.

PROGRAM:

```
echo enter the year:
read y
c=$((y%4))
if [ $c -eq 0 ]
then
```

```
    echo IT IS A LEAP YEAR
else
echo IT IS NOT A LEAP YEAR
fi
```

OUTPUT:



```
enter the year:
2004
IT IS A LEAP YEAR
```

d) To find the factorial of a given number using white and for loop

AIM:

To find the factorial of a given number using while and for loop

i) Using while loop

ALGORITHM:

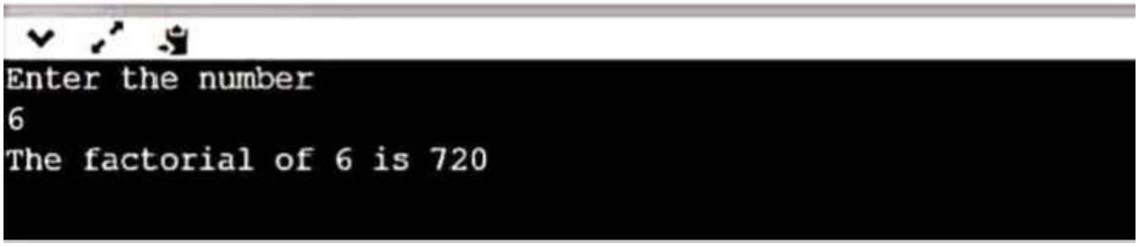
1. Start
2. Declare the number to find factorial
3. Read the number, Declare factorial = 1
4. Using while loop, if the number if more than o
5. Then the factorial is found by fact = $(expr $face/$n). n=$(n-1) expression. I
6. End.

PROGRAM:

```
echo Enter th   number
read num
n=$num
fact=1
while [Sn-gt0]
    do
```

```
      n=$((n-1))
  done
echo The factorial of $num is $fact
```

OUTPUT:



```
Enter the number
6
The factorial of 6 is 720
```

ii) Using for loop:

ALGORITHM:

1.Start

2. Declare the number to find factorial

3. Read the number, Declare factorial

4. using for loop assign i=1; i<=num, itt (in for loop)

5. fact if assigned as $ (expr $ fact 1 + $1)

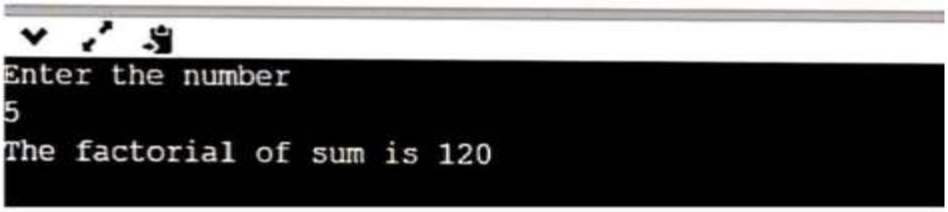
6. factorial can be found.

7. End.

PROGRAM:

```
echo Enter th   number
read num
fact=1
for ((i=1;i<=num;i++))
  do
      fact=$(expr $fact \* $i)
  done
echo The factorial of sum is $fact
```

OUTPUT:



```
Enter the number
5
The factorial of sum is 120
```

e) To find the fibonacci of a given numbers using while loop

AIM:

To find the fibonacci of a given numbers using while loop

ALGORITHM:

1. Start

2. Get a number

3. Use white loop to compute the fibonacci by using the below formula.

4. fibo = $((a+b))

5. Swap a = $b,b= $+n

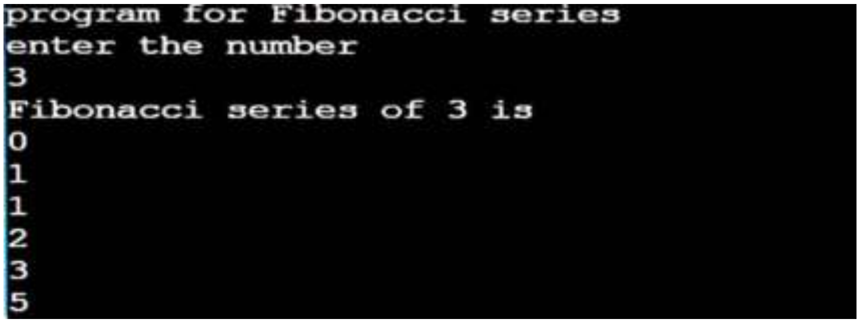
6. End.

PROGRAM:

```
echo program for Fibonacci series

echo enter the number

read n

f=1

j=0

echo Fibonacci series of $n is

echo $j

echo $f

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

do

k=expr $j \+ $f

j=$f

f=$k

echo $k
```

```
done
```

OUTPUT:



```
program for Fibonacci series
enter the number
3
Fibonacci series of 3 is
0
1
1
2
3
5
```

f) To find the greatest of three numbers using if elif...if

AIM:

To find the greatest of s numbers using if elif….if

ALGORITHM:

1. Start

2. Declare 3 numbers.

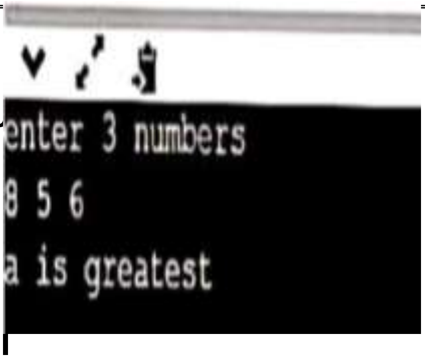
3. if 'a' is greater than 'b' and 'c', 'a' is larger else if 'b' is more than 'a' and 'c', 'b' is greater else i 'c' is more than 'a' and 'b','c' is greater

6. End.

PROGRAM:

```
echo enter 3 numbers
read a b c
if [ $a -gt $b a $a -gt $c ]
then
echo a is greatest
elif [ $b -gt $c ]
then
echo bis greatest
else
echo c is greatest
fi
```

OUTPU



```
enter 3 numbers
8 5 6
a is greatest
```

RESULT

Thus the program using control structures in shell programming was compiled and executed successfully.

Ex No : 3	Use of CASE statements and Menu Driven Program
07.05.2024	

AIM:

To illustrate the use of switch statement in Menu driven using Shell Programming

ALGORITHM:

```
1. Start
2. Read the choice from the user
3. Use switch case statement to do the operation
        3.1 If choice is 1 perform Fibonacci series
        3.2 If choice is 2 write a shell program to check whether a given number is odd or even
        3.3 If choice is 3, write a shell program to check whether a given year is Leap year or not
        3.4 If choice is 4 write a shell program to find the greatest of three numbers
        3.5 If choice is 5, write a shell program to find the sum of the digits of a given number
```

2. Menu Driven program for file operations:

```
1. Start
2. Using while loop, perform
3. Get the choice from user
4. Create switch case to perform
        4.1 If choice is 1, perform cat command
        4.2 If choice is 1, perform cp command
        4.3 If choice is 3, perform mv command
        4.4 If choice is 4, perform wc command
        4.5 If choice is 5, perform grep command
        4.6 If choice is 6, perform head command
        4.7 If choice is 7, perform tail command
        4.8 If choice is 8, perform sort command
```

PROGRAM:

```
echo "MENU"
echo 1. Fibonacci
echo 2. Odd or Even

echo 3. Leap Year

echo 4. Greatest of Three Numbers

echo 5. Sum of Digits

echo Enter your choice:

read choice

case $choice in

1) echo Program for Fibonacci series

echo Enter the number:

read n

f=1

j=0

echo Fibonacci series of $n is

echo $j

echo $f

for (( i=2; i<n; i++ ))

do

k=$((j + f))

j=$f

f=$k

echo $k

done

;;

2) echo Program for odd or even number

echo Enter the number:

read n

a=`expr $n % 2`

if [ $a -eq 0 ]

then
```

```
echo "The given number $n is even number"
else
echo "The given number $n is odd number"
fi
;;
3) echo program to find the given year is leap or not
echo Enter a year you want to find
read a
b=`expr $a % 4`
if [ $b -eq 0 ]
then
echo "The year $a is a leap year"
else
echo "The year $a is not a leap year"
fi
;;
4) echo Enter the numbers:
read a b c
if [ $a -gt $b ] && [ $a -gt $c ]
then
echo "The number $a is greater"
elif [ $b -gt $a ] && [ $b -gt $c ]
then
echo "The number $b is greater"
else
echo "The number $c is greater"
fi
;;
5) echo Enter a number:
```

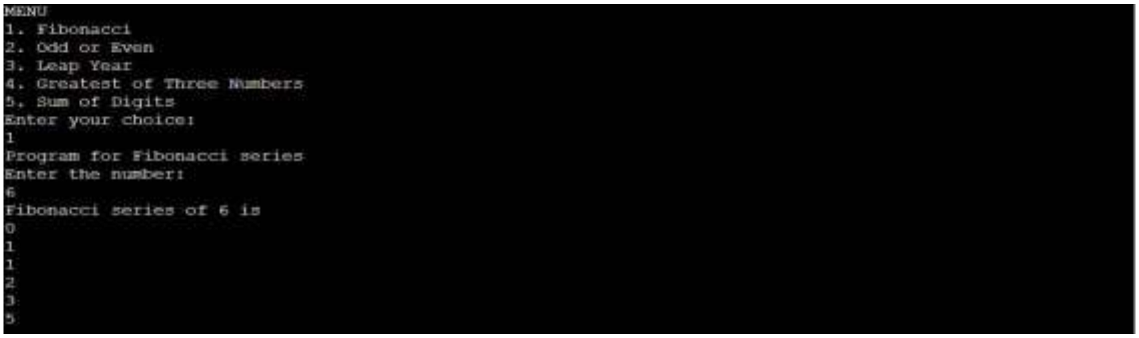
```
read a
i=$a
k=0
while [ $a -gt 0 ]
do
digit=$((a % 10))
sum=$((sum + digit))
a=$((a / 10))
done
echo "Sum of $i is $sum"
;;
Esac
2.while(true)
do
echo MENU
echo 1. cat
echo 2. cp
echo 3. mv
echo 4. wc
echo 5. grep -option
echo 6. head -n
echo 7. tail -n
echo 8. sort -option
echo Enter your choice
read choice
case $choice in
1) echo Enter the file name
read file
cat $file
```

```
;;
2) echo Enter source file name
read s
echo Enter destination file name
read d
cp $s $d
;;
3) echo Enter source file name
read s
echo Enter destination file name
read d
mv $s $d
;;
4) echo Enter the file name
read file
wc $file
;;
5) echo Enter the file name
read file
echo Enter the option v,n,c
read option
echo Enter the word
read word
grep -$option $word $file
;;
6) echo Enter the file name
read file
echo Enter no. of lines to display
read n
```

```
head -$n $file

;;

7) echo Enter the file name

read file

echo Enter no. of lines to display

read n

tail -$n $file

;;

8) echo Enter the file name

read file

echo Enter the option

read option

sort -$option $file

;;

*)echo Enther the correct choice

;;

esac

done
```

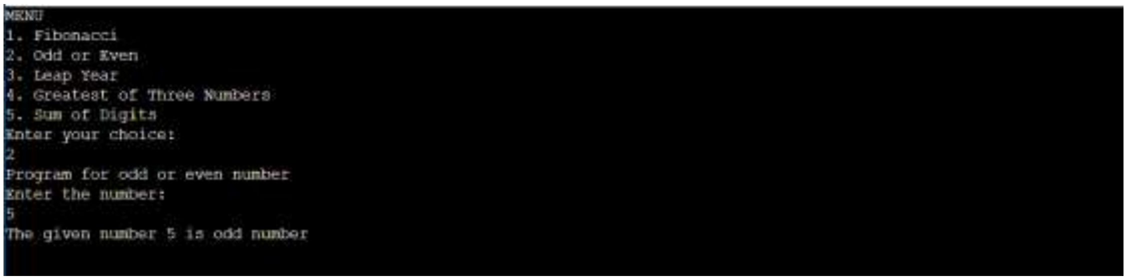
OUTPUT: MENU DRIVEN PROGRAMMING

1. FIBONACCI



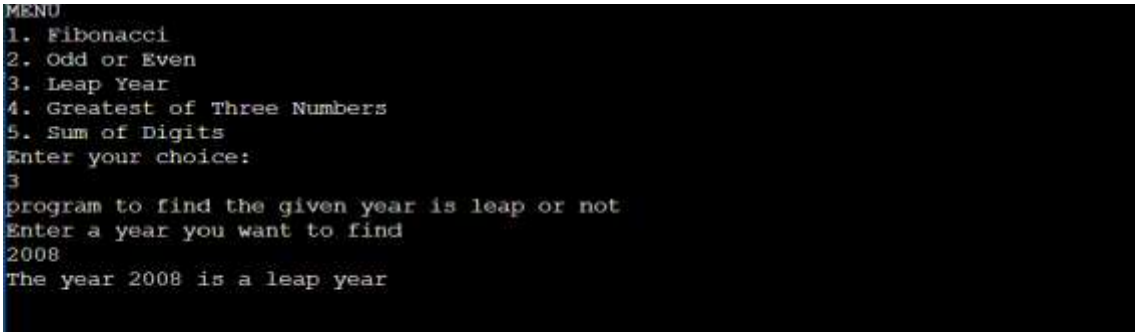
```
MENU
1. Fibonacci
2. Odd or Even
3. Leap Year
4. Greatest of Three Numbers
5. Sum of Digits
Enter your choice:
1
Program for Fibonacci series
Enter the number:
6
Fibonacci series of 6 is
0
1
1
2
3
5
```

2. ODD OR EVEN NUMBER



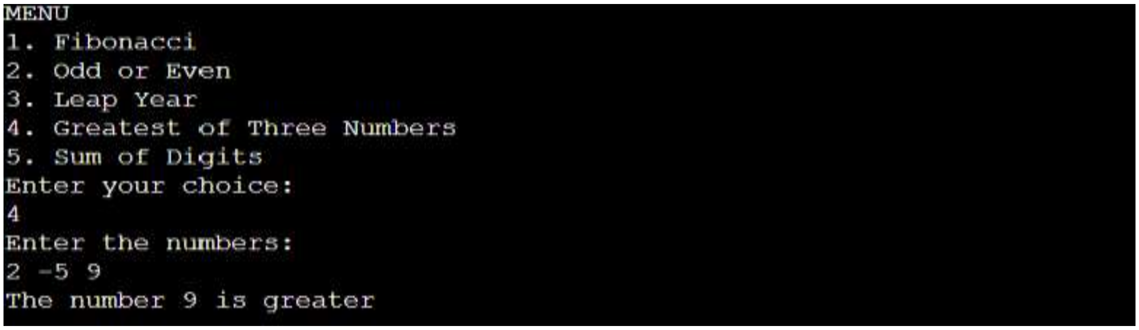
```
MENU
1. Fibonacci
2. Odd or Even
3. Leap Year
4. Greatest of Three Numbers
5. Sum of Digits
Enter your choice:
2
Program for odd or even number
Enter the number:
5
The given number 5 is odd number
```

3. LEAP YEAR



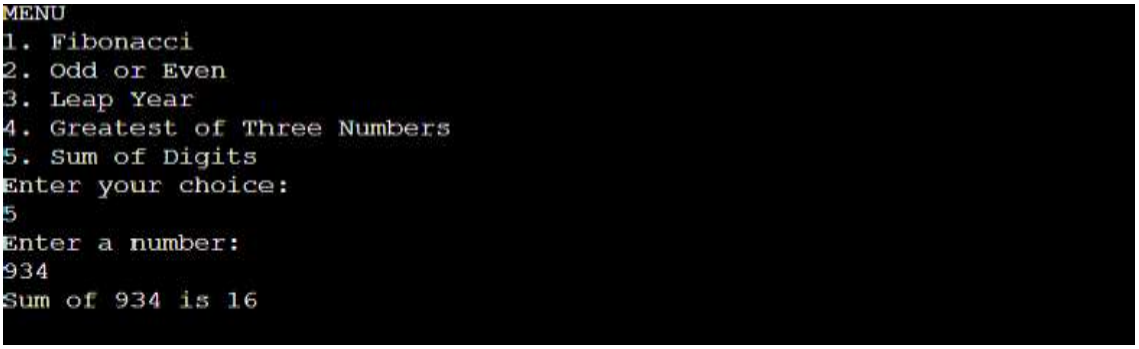
```
MENU
1. Fibonacci
2. Odd or Even
3. Leap Year
4. Greatest of Three Numbers
5. Sum of Digits
Enter your choice:
3
program to find the given year is leap or not
Enter a year you want to find
2008
The year 2008 is a leap year
```

4. GREATEST OF THREE NUMBERS



```
MENU
1. Fibonacci
2. Odd or Even
3. Leap Year
4. Greatest of Three Numbers
5. Sum of Digits
Enter your choice:
4
Enter the numbers:
2 -5 9
The number 9 is greater
```

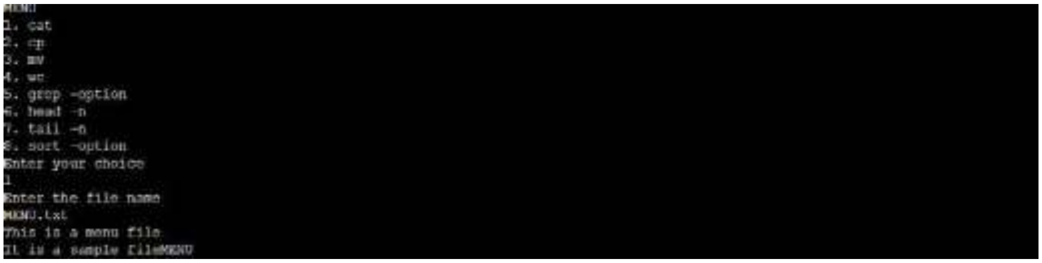
5. SUM OF DIGITS



```
MENU
1. Fibonacci
2. Odd or Even
3. Leap Year
4. Greatest of Three Numbers
5. Sum of Digits
Enter your choice:
5
Enter a number:
934
Sum of 934 is 16
```

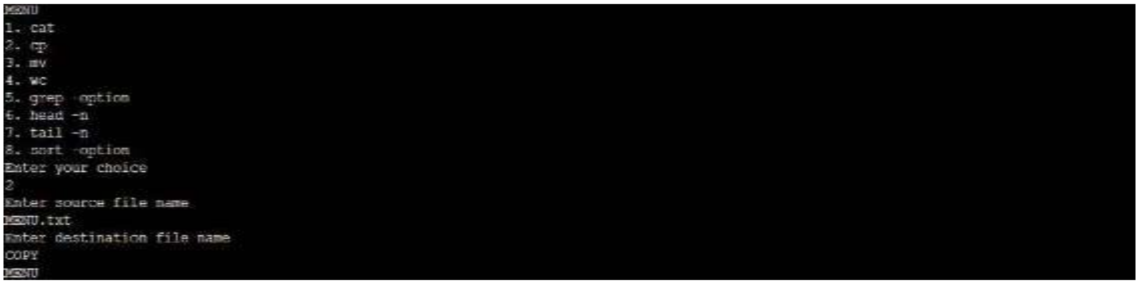
OUTPUT:

1.CAT



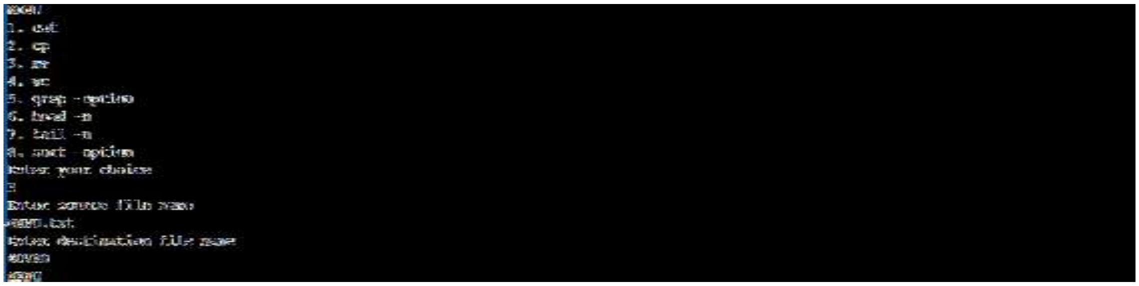
```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
1
Enter the file name
MENU.txt
This is a menu file
It is a sample fileMENU
```

2. CP



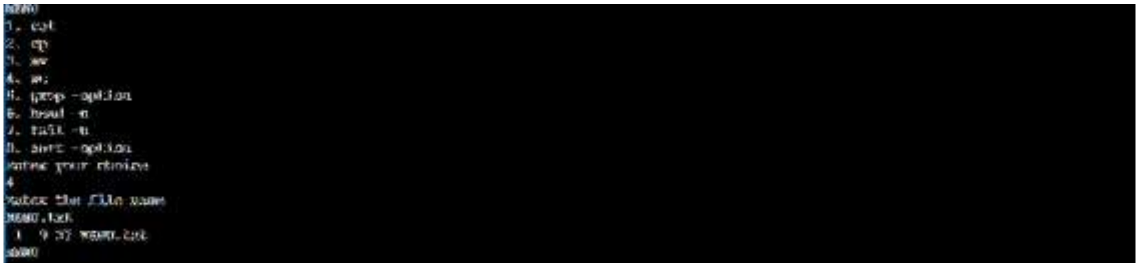
```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
2
Enter source file name
MENU.txt
Enter destination file name
COPY
MENU
```

3. MV



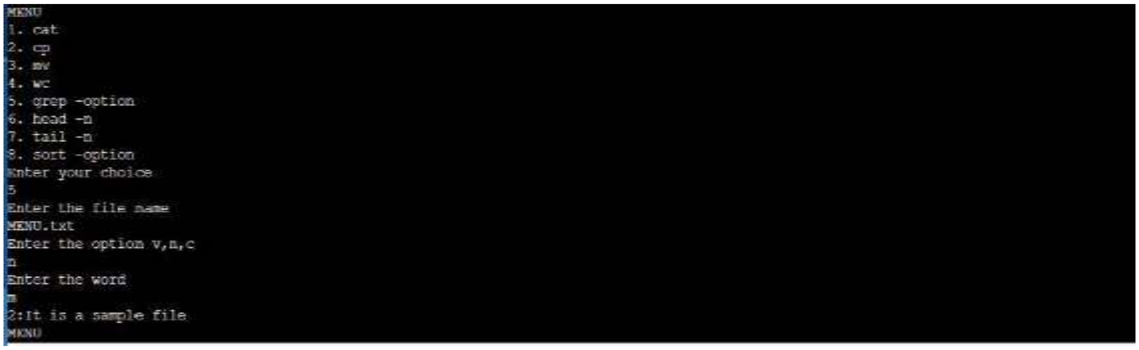
```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
3
Enter source file name
MENU.txt
Enter destination file name
MOVED
MENU
```

4. WC



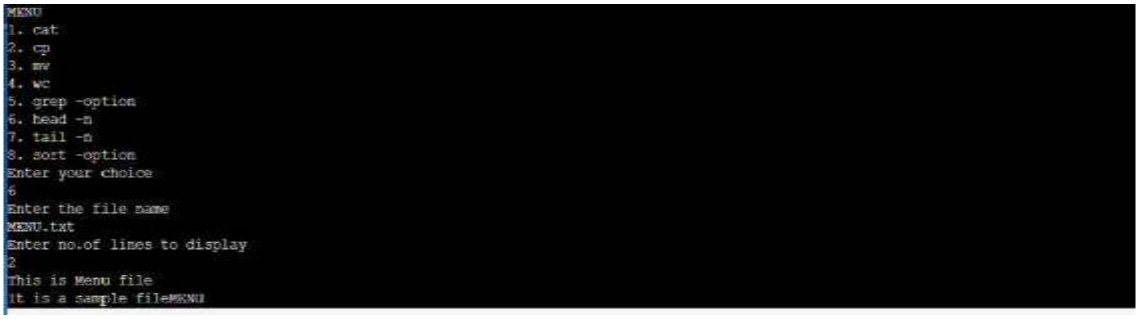
```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
4
Enter the file name
MENU.txt
 1  9 37 MENU.txt
MENU
```

5. GREP



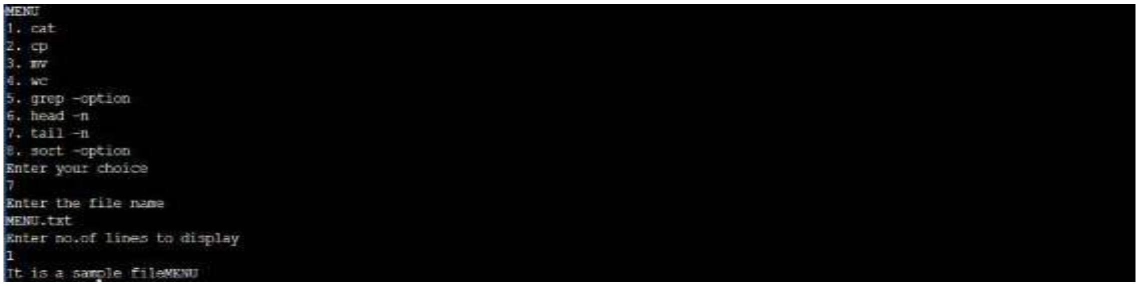
```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
5
Enter the file name
MENU.txt
Enter the option v,n,c
n
Enter the word
m
2:It is a sample file
MENU
```

6. HEAD



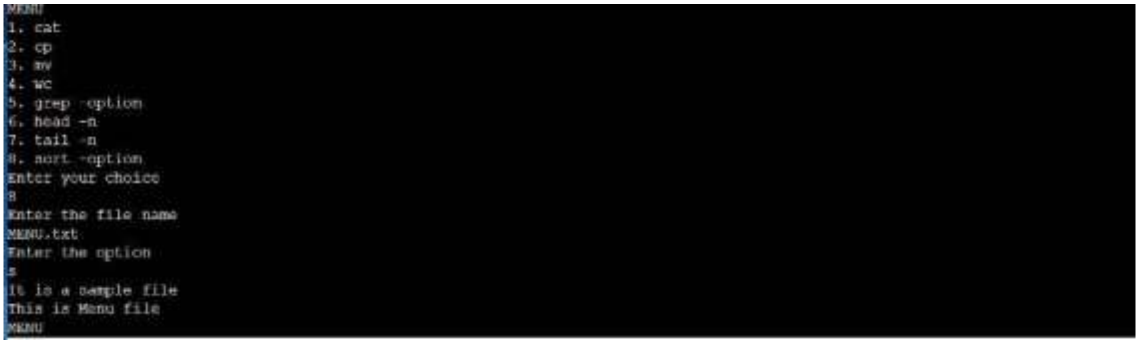
```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
6
Enter the file name
MENU.txt
Enter no.of lines to display
2
This is Menu file
It is a sample fileMENU
```

7. TAIL



```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
7
Enter the file name
MENU.txt
Enter no.of lines to display
1
It is a sample fileMENU
```

8. SORT



```
MENU
1. cat
2. cp
3. mv
4. wc
5. grep -option
6. head -n
7. tail -n
8. sort -option
Enter your choice
8
Enter the file name
MENU.txt
Enter the option
s
It is a sample file
This is Menu file
MENU
```

RESULT:

The following menu driven program was compiled and executed successfully.

Ex No : 4	**Implementation of FCFS and SJF CPU scheduling algorithms**
08.05.2024	

AIM

To write a C++ program to implement the FIFO (FCFS) and SJF CPU Scheduling algorithms.

Algorithm:

1. Start the program.
2. Declare the variable.
3. Input the number of processes from user.
4. Create ‘for’ loop to input arrival time and burst time.
5. Using ‘for’ loop, calculate:
    - i. Turn Around Time = Completion Time – Arrival Time
    - ii. Waiting Time =Turn Around Time – Burst Time
6. Calculate Average Turn Around Time and Average Waiting time.
7. Print the results.
8. Stop the program.

Program:

<u>CODE for FIFO</u>

```
// C++ program for implementation of FCFS scheduling


#include<iostream>
using namespace std;

void findWaitingTime(int processes[], int n,
                     int bt[], int wt[])
```

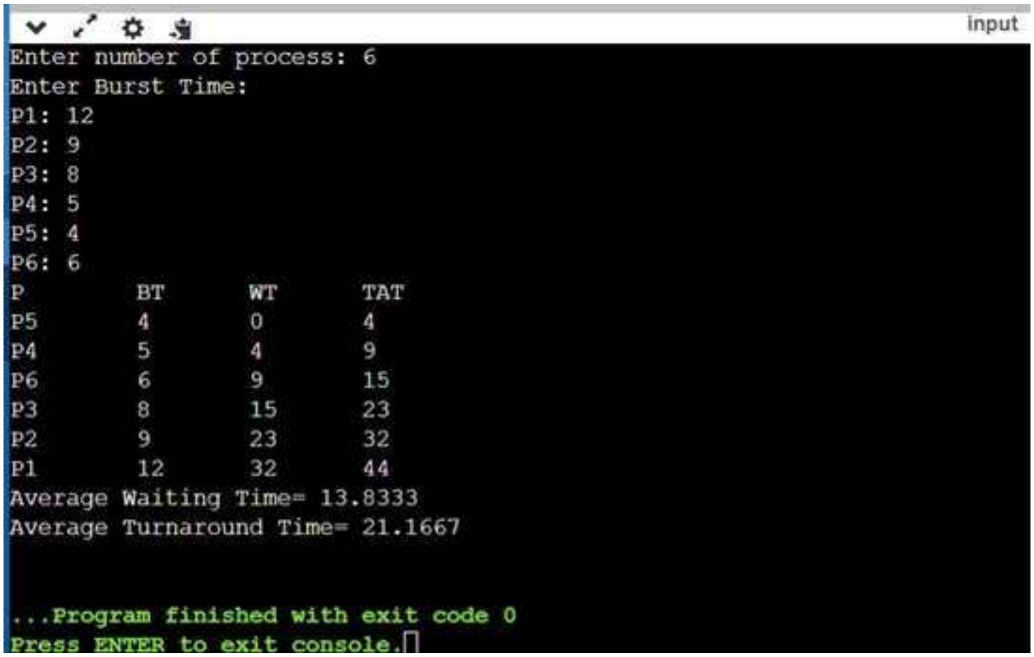
```
{
    wt[0] = 0;
    for (int  i = 1; i < n ; i++ )
        wt[i] =  bt[i-1] + wt[i-1] ;
}

void findTurnAroundTime( int processes[], int n,
                  int bt[], int wt[], int tat[])
{
    for (int  i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

void findavgTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    findWaitingTime(processes, n, bt, wt);
     findTurnAroundTime(processes, n, bt, wt, tat);
     cout << "Processes  "<< " Burst time  " << " Waiting time  " << " Turn around time\n";
     for (int  i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << "   " << i+1 << "\t\t" << bt[i] <<"\t     " << wt[i] <<"\t\t  " << tat[i] <<endl;
    }
     cout << "Average waiting time = " << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
}

int main()
{
int A,B,C,D,E;
cout<<"ENTER JOBS"<<endl;
cin >>A>>B>>C>>D>>E;
int job[] = { A, B, C, D, E};
int n = sizeof job / sizeof job[0];
int B1,B2,B3,B4,B5;
cout<<"ENTER  BURST  TIME"<<endl;

cin>>B1>>B2>>B3>>B4>>B5;
int burst_time[] = {B1, B2, B3, B4, B5};
findavgTime(job, n, burst_time);
return 0;
}
```

CODE for SJF:

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

int main() {

        int A[100][4];
        int i, j, n, total = 0, index, temp;
        float avg_wt, avg_tat;
        cout << "Enter number of process: ";
        cin >> n;
        cout << "Enter Burst Time:" << endl;
                for (i = 0; i < n; i++) {
                cout << "P" << i + 1 << ": ";
                cin >> A[i][1];
                A[i][0] = i + 1;
        }

        for (i = 0; i < n; i++) {
                index = i;
                for (j = i + 1; j < n; j++)
                        if (A[j][1] < A[index][1])
                                index = j;
                temp = A[i][1];
                A[i][1] = A[index][1];
                A[index][1] = temp;
                temp = A[i][0];
                A[i][0] = A[index][0];
                A[index][0] = temp;
        }

        A[0][2] = 0;

        for (i = 1; i < n; i++) {
                A[i][2] = 0;
                for (j = 0; j < i; j++)
                        A[i][2] += A[j][1];
                total += A[i][2];
        }

        avg_wt = (float)total / n;
        total = 0;
        cout << "P       BT      WT      TAT" << endl;

        for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
        cout << "P" << A[i][0] << "       " << A[i][1] << "    " << A[i][2] << "      " << A[i][3] << endl;
        }
        avg_tat = (float)total / n;
        cout << "Average Waiting Time= " << avg_wt << endl;
        cout << "Average Turnaround Time= " << avg_tat << endl;
}
```

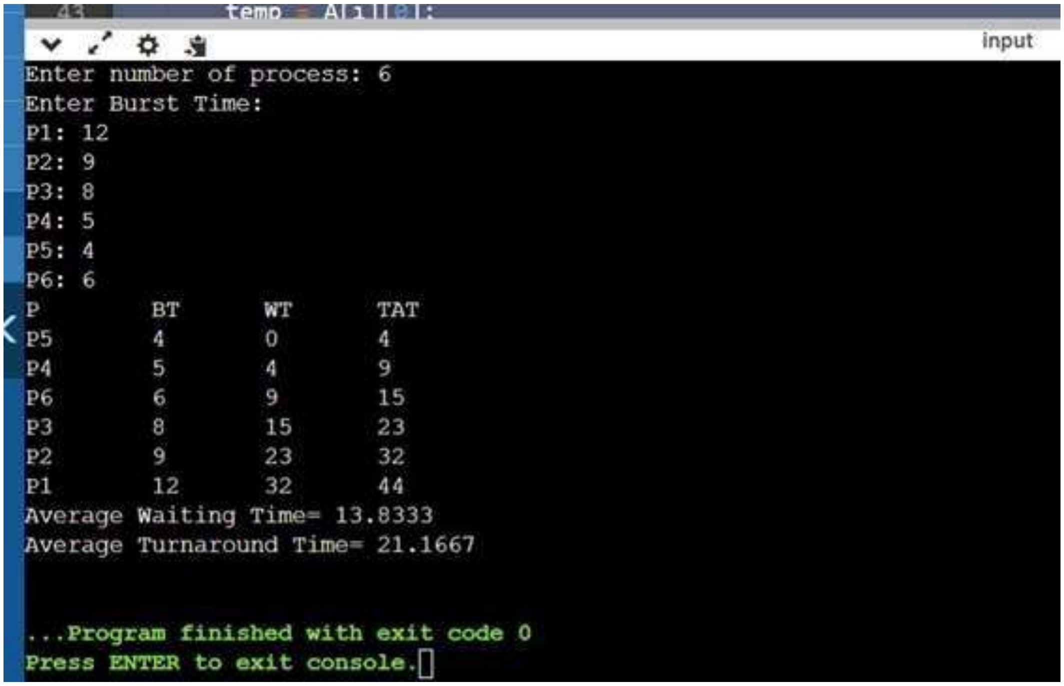
Output:

FIFO:



```
Enter number of process: 6
Enter Burst Time:
P1: 12
P2: 9
P3: 8
P4: 5
P5: 4
P6: 6
P        BT       WT       TAT
P5       4        0        4
P4       5        4        9
P6       6        9        15
P3       8        15       23
P2       9        23       32
P1       12       32       44
Average Waiting Time= 13.8333
Average Turnaround Time= 21.1667


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

SJF:



```
Enter number of process: 6
Enter Burst Time:
P1: 12
P2: 9
P3: 8
P4: 5
P5: 4
P6: 6
P        BT       WT       TAT
P5       4        0        4
P4       5        4        9
P6       6        9        15
P3       8        15       23
P2       9        23       32
P1       12       32       44
Average Waiting Time= 13.8333
Average Turnaround Time= 21.1667


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

RESULT

The average turnaround time and average waiting time is calculated successfully using FIFO and SJF CPU scheduling algorithm.

Ex No : 5	Implementation of Priority CPU scheduling Algorithm
09.05.2024	

AIM:

To write a program for the Priority and Round Robin CPU Scheduling.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the required arrays.

Step 3: Get the input of burst time, arrival time, priority from the user.

Step 4: Compute the required calculations like

- Completion Time (CT)
- Turn Around Time (TAT)
- Waiting Time (WT)

using the required formula.

Step 5: Print the required output.

Step 6: Stop the program.

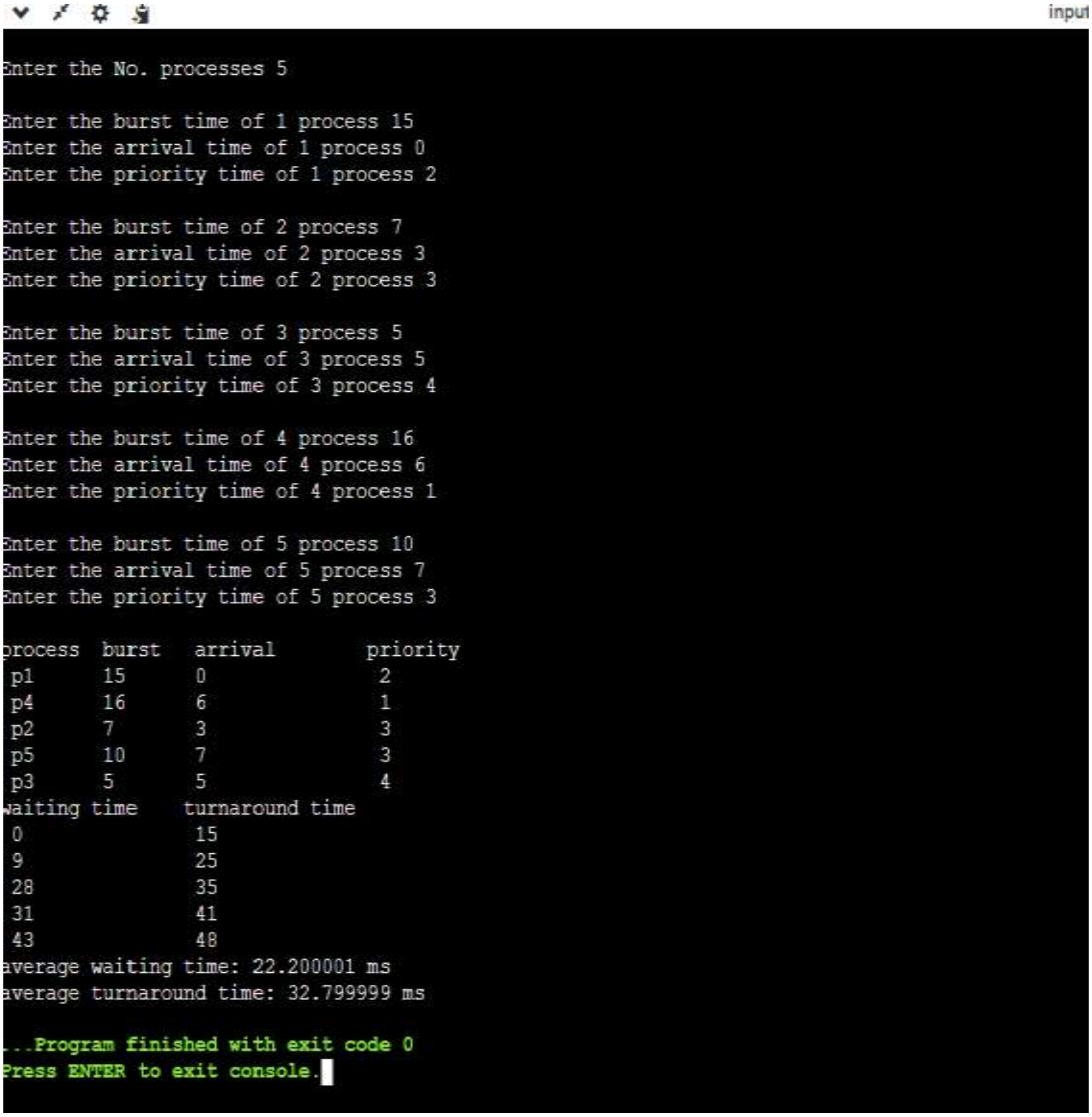
CODE:

```
#include<stdio.h>
int main() {
  int i, n, p[10] = {1,2,3,4,5,6,7,8,9,10}, min, k=1, burst=0, pri[100];
  int bt[100], temp, temp1, j, at[1000], wt[1000], rt[1000], tt[1000], ta=0,sum=0;
  float wavg, tavg, tsum, wsum;
  printf("\nEnter the No. processes ");
  scanf("%d", &n);
  for(i=0; i<n; i++) {
    printf("\nEnter the burst time of %d process ", i+1);
    scanf("%d", &bt[i]);
    printf("Enter the arrival time of %d process ", i+1);
    scanf("%d", &at[i]);
    printf("Enter the priority time of %d process ", i+1);
    scanf("%d", &pri[i]);
  }
  for(i=0; i<n; i++) {
    for(j=0; j<n;j ++) {
      if(at[i] < at[j]) {
        temp = p[j];
        p[j] = p[i];
        p[i] = temp;
        temp = at[j];
        at[j] = at[i];
        at[i] = temp;
        temp1 = bt[j];
        bt[j] = bt[i];
```

```
            bt[i] = temp1;
        }
    }
}
for(j=0; j<n; j++) {
    burst = burst+bt[j];
    min = bt[k];
    for(i=k; i<n; i++) {
        min = pri[k];
        if(burst >= at[i]) {
            if(pri[i] < min) {
                temp = p[k];
                p[k] = p[i];
                p[i] = temp;
                temp = at[k];
                at[k] = at[i];
                at[i] = temp;
                temp1 = bt[k];
                bt[k] = bt[i];
                bt[i] = temp1;
                temp = pri[k];
                pri[k] = pri[i];
                pri[i] = temp;
            }
        }
    }
    k++;
}
wt[0] = 0;
for(i=1; i<n; i++) {
    sum = sum+bt[i-1];
    wt[i] = sum-at[i];
}
for(i=0; i<n; i++) {
    wsum = wsum+wt[i];
}
wavg = wsum/n;
for(i=0; i<n; i++) {
    ta = ta+bt[i];
    tt[i] = ta-at[i];
}
for(i=0; i<n; i++) {
    tsum = tsum+tt[i];
}
tavg = tsum/n;
for(i=0; i<n; i++) {
    rt[i] = wt[i];
}
```

```
    printf("\nprocess\t burst\t arrival\tpriority ");
    for(i=0; i<n; i++) {
      printf("\n p%d", p[i]);
      printf("\t %d", bt[i]);
      printf("\t %d ", at[i]);
      printf("\t\t %d", pri[i]);
    }
    printf("\nwaiting time\tturnaround time");
    for(i=0; i<n; i++) {
      printf("\n %d", wt[i]);
      printf("\t\t %d",tt[i]);
    }
    printf("\naverage waiting time: %f ms", wavg);
    printf("\naverage turnaround time: %f ms", tavg);
}
```

OUTPUT:



```
Enter the No. processes 5

Enter the burst time of 1 process 15
Enter the arrival time of 1 process 0
Enter the priority time of 1 process 2

Enter the burst time of 2 process 7
Enter the arrival time of 2 process 3
Enter the priority time of 2 process 3

Enter the burst time of 3 process 5
Enter the arrival time of 3 process 5
Enter the priority time of 3 process 4

Enter the burst time of 4 process 16
Enter the arrival time of 4 process 6
Enter the priority time of 4 process 1

Enter the burst time of 5 process 10
Enter the arrival time of 5 process 7
Enter the priority time of 5 process 3

process  burst   arrival        priority
 p1      15      0               2
 p4      16      6               1
 p2      7       3               3
 p5      10      7               3
 p3      5       5               4
waiting time    turnaround time
 0               15
 9               25
 28              35
 31              41
 43              48
average waiting time: 22.200001 ms
average turnaround time: 32.799999 ms

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

Ex No : 6	C Simulation of vi, cp and cat commands
10.05.2024	

RESULT:

The above program for implementing priority CPU scheduling algorithms was compiled and executed successfully.

AIM:

To write a c program for simulatión of vi, cat and Cp commands

ALGORITHM:

- Start the program and initialize Variables
- Display Menu and get User Choice
- Switch on User Choice
- Case 1 - vi Command
  - Print Termination Instruction
  - Open File for Writing (vi Command)
  - Read and Write Characters to File
- Case 2 - cat Command
  - Open File for Reading (cat Command)
  - Read and Display File Content
- Case 3 - cp Command
  - Open Source and Destination Files (cp Command), Copy File Content
- End the program

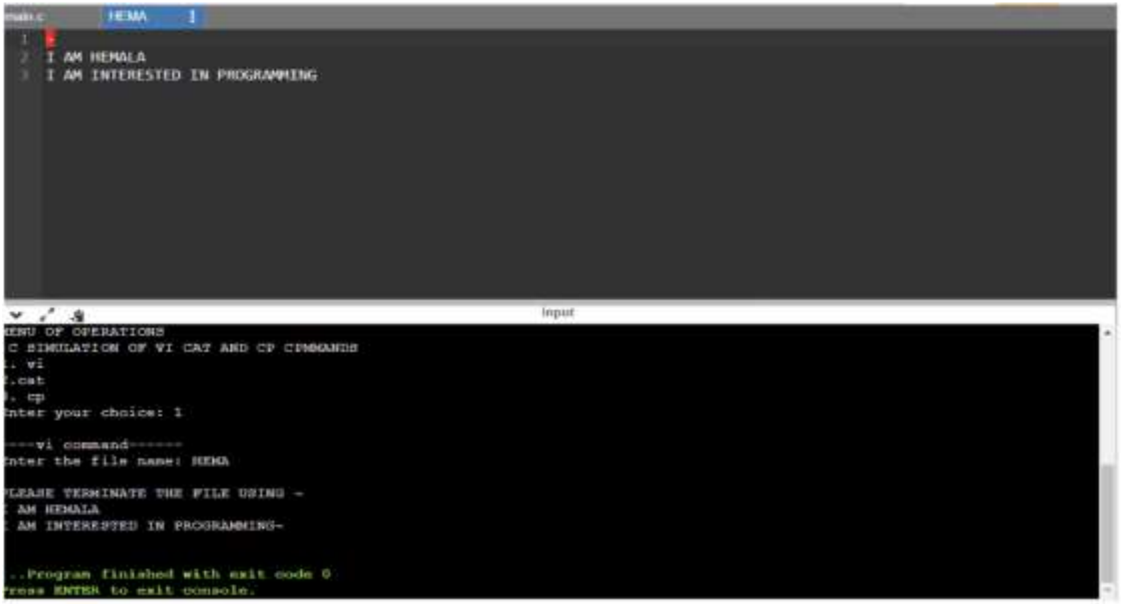
PROGRAM:

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

int ch;

char a,b,fn1[10],fn2[10];
```

```
FILE *f1,*f2;

printf("MENU OF OPERATIONS\n C SIMULATION OF VI CAT AND CP COMMANDS ");

printf("\n1. vi \n2.cat \n3. cp ");

printf("\nEnter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

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

printf("\nEnter the file name: ");

scanf("%s",fn1);

printf("\n PLEASE TERMINATE THE FILE USING ~ ");

printf("\n");

f1=fopen(fn1,"w");

while(a!='~')

{

fputc(a,f1);

a=getchar();

}

fclose(f1);

break;

case 2:

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

printf("\n Enter the file name: ");

scanf("%s",fn1);

f1=fopen(fn1,"r");

if(f1=='\0')

{

printf("\n File is empty");

exit(0);
```

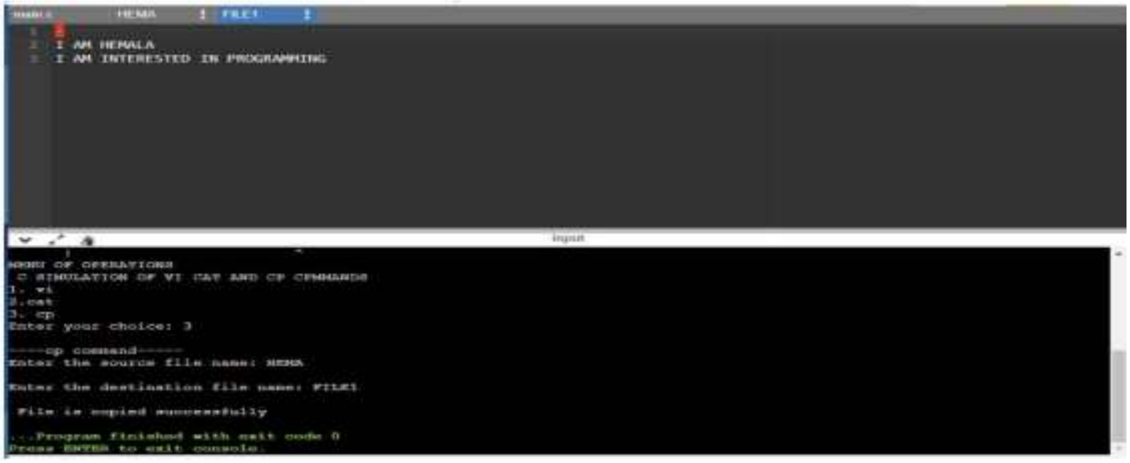
```
}
else
{
a=fgetc(f1);
while(a!=EOF)
{
 printf("%c",a);
a=fgetc(f1);
}
}
fclose(f1);
break;
case 3:
printf("\n----cp  command -----");
printf("\nEnter the source file name: ");
scanf("%s",fn1);
printf("\nEnter the destination file name: ");
scanf("%s",fn2);
f1=fopen(fn1,"r");
f2=fopen(fn2,"w");
if(f1=='\0' && f2=='\0')
{
printf("\nFile is empty");
}
else
{
b=fgetc(f1);
while(b!=EOF)
{
fputc(b,f2);
b=getc(f1);
}  }
fclose(f1);  fclose(f2);
printf("\n File is copied successfully");
break;
default:
printf("\nEnter a valid option");
} }
```

OUTPUT:



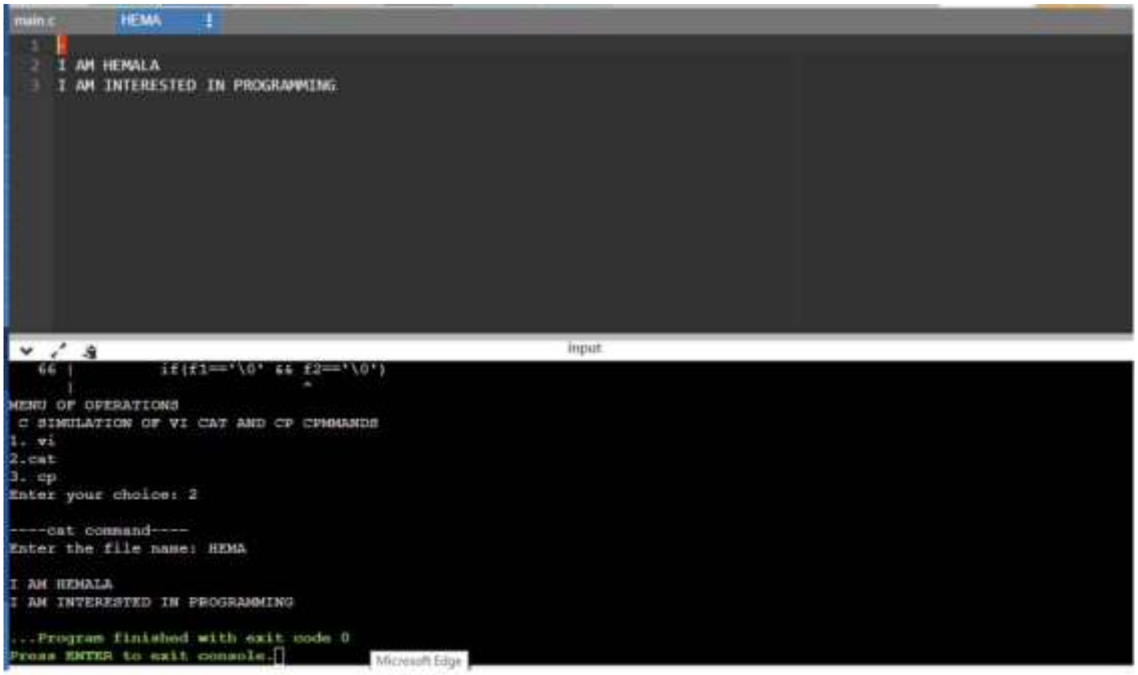
```

I AM HEMALA
I AM INTERESTED IN PROGRAMMING
```

```
ENU OF OPERATIONS
 C SIMULATION OF VI CAT AND CP CPMMANDS
. vi
.cat
. cp
nter your choice: 1

----vi command------
nter the file name: HEMA

LEASE TERMINATE THE FILE USING ~
 AM HEMALA
 AM INTERESTED IN PROGRAMMING~


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



```

I AM HEMALA
I AM INTERESTED IN PROGRAMMING
```

```
MENU OF OPERATIONS
 C SIMULATION OF VI CAT AND CP CPMMANDS
1. vi
2.cat
3. cp
Enter your choice: 3

-----cp command-----
Enter the source file name: HEMA

Enter the destination file name: FILE1

 File is copied successfully

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



```

I AM HEMALA
I AM INTERESTED IN PROGRAMMING
```

```
   66 |         if(f1=='\0' && f2=='\0')
      |                                ^
MENU OF OPERATIONS
 C SIMULATION OF VI CAT AND CP CPMMANDS
1. vi
2.cat
3. cp
Enter your choice: 2

----cat command----
Enter the file name: HEMA

I AM HEMALA
I AM INTERESTED IN PROGRAMMING

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

RESULT:

Thus the above program to simulate the vi, cat and cp commands was compiled and executed successfully.

Ex No : 7	Use of process ,file, stat and directory system calls
11.05.2024	

Aim:

To write a C++ program to implement the stimulation of vi, cat and cp commands.

Algorithm:

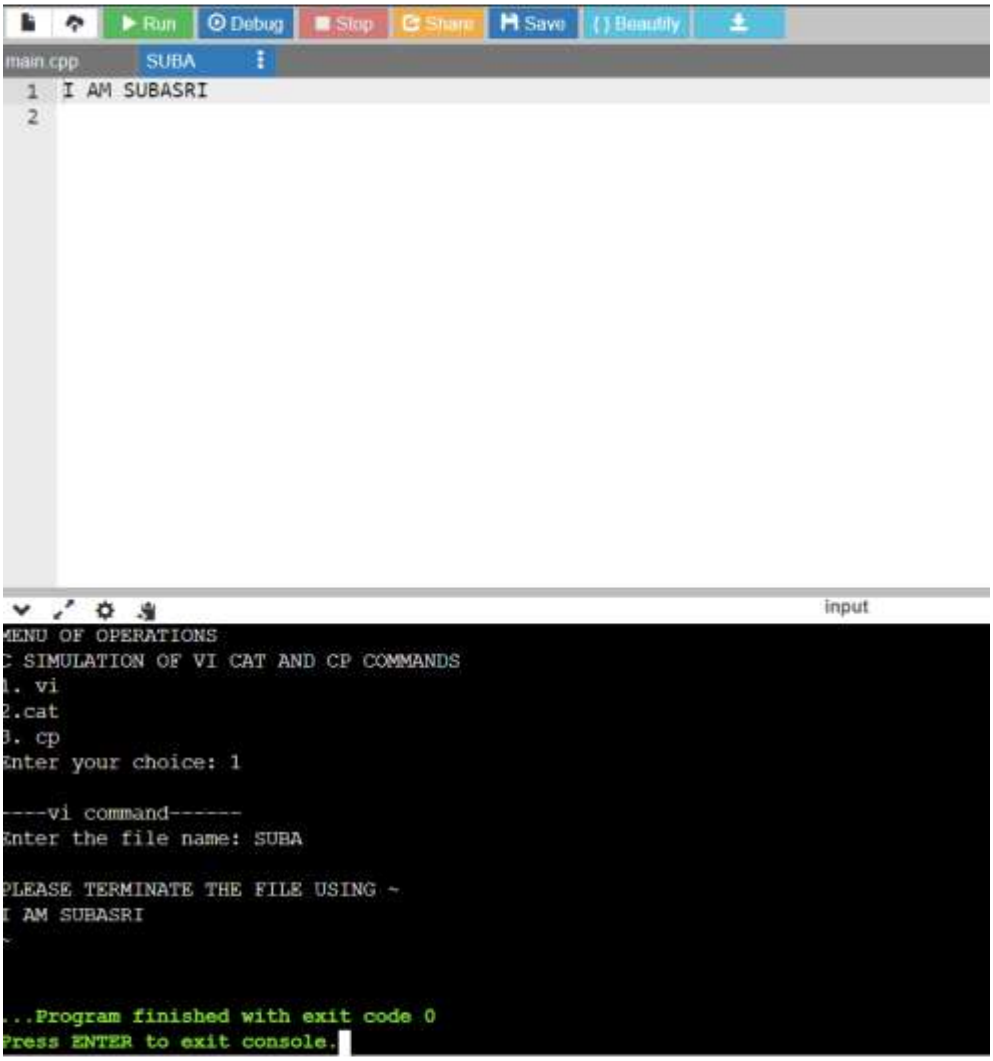
1. Start the execution of the program.
2. Include the required header files.
3. Declare file pointers.
4. Use ‘switch case’ to get the choice from the user.
5. Get file name as input from the user.
6. Execute the required commands.
7. Stop the program.

Program:

```
#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

int main()
{
   int ch;
   char a, b;
   string fn1, fn2;
   fstream f1, f2;

   cout << "MENU OF OPERATIONS\nC SIMULATION OF VI CAT AND CP
COMMANDS ";
   cout << "\n1. vi \n2.cat \n3. cp ";
   cout << "\nEnter your choice: ";
   cin >> ch;

   switch (ch)
   {
   case 1:
      cout << "\n----vi command------";
      cout << "\nEnter the file name: ";
```

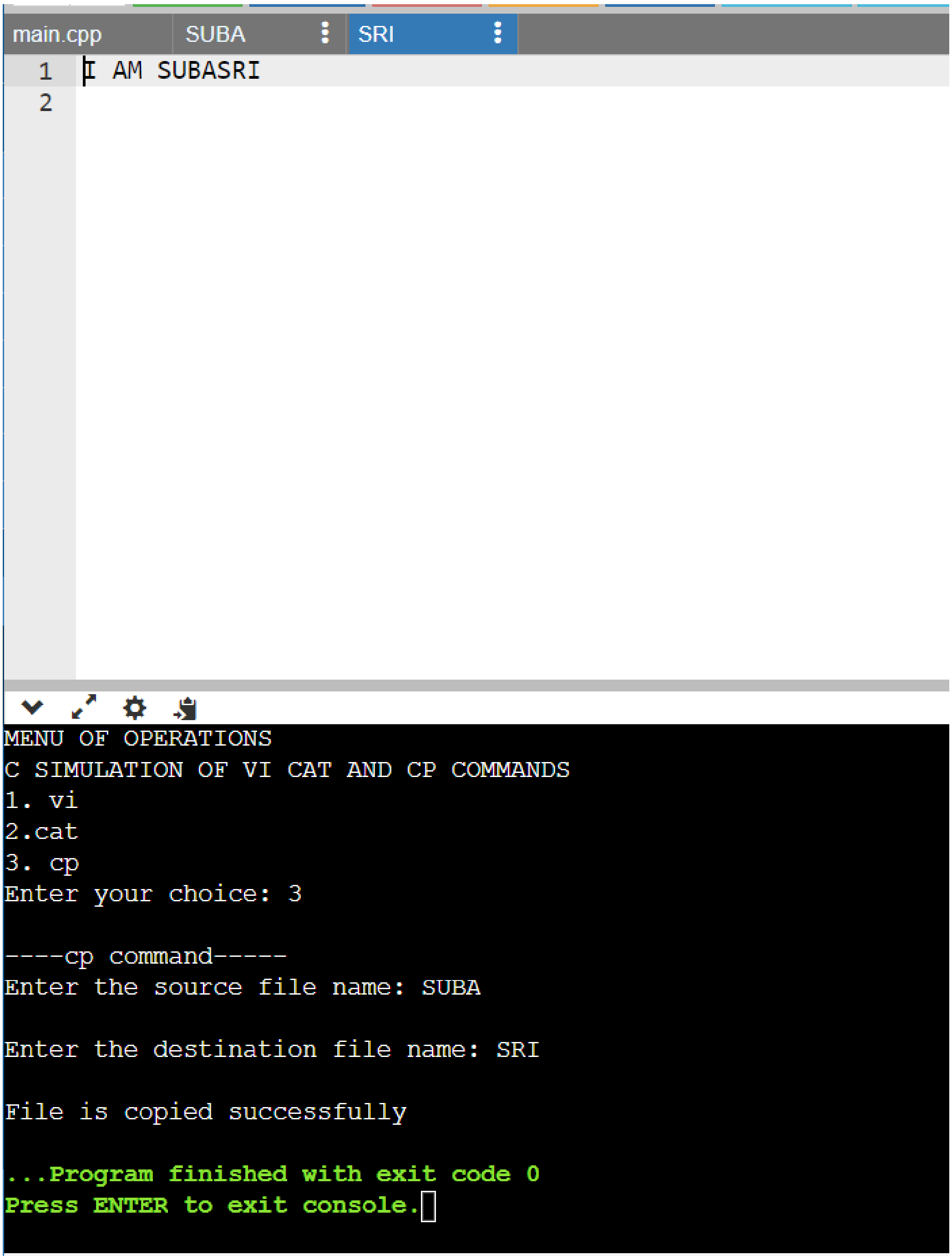
```
        cin >> fn1;
        cout << "\nPLEASE TERMINATE THE FILE USING ~ ";
        cout << "\n";
        f1.open(fn1, ios::out);
        if (!f1) {
            cerr << "Error opening file for writing" << endl;
            return 1;
        }
        cin.ignore(); // To ignore the newline character left in the input buffer
        while (cin.get(a) && a != '~') {
            f1.put(a);
        }
        f1.close();
        break;

    case 2:
        cout << "\n----cat command----";
        cout << "\nEnter the file name: ";
        cin >> fn1;
        f1.open(fn1, ios::in);
        if (!f1) {
            cout << "\nFile does not exist or cannot be opened";
            return 1;
        } else {
            while (f1.get(a)) {
                cout << a;
            }
        }
        f1.close();
        break;

    case 3:
        cout << "\n----cp command-----";
        cout << "\nEnter the source file name: ";
        cin >> fn1;
        cout << "\nEnter the destination file name: ";
        cin >> fn2;
        f1.open(fn1, ios::in);
        f2.open(fn2, ios::out);
        if (!f1) {
            cout << "\nSource file does not exist or cannot be opened";
            return 1;
        } else if (!f2) {
            cout << "\nDestination file cannot be opened for writing";
            return 1;
        } else {
            while (f1.get(b)) {
                f2.put(b);
            }
        }
        f1.close();
        f2.close();
        cout << "\nFile is copied successfully";
        break;

    default:
        cout << "\nEnter a valid option";
    }

    return 0;
}
```

Output:



```
I AM SUBASRI
```

```
MENU OF OPERATIONS
C SIMULATION OF VI CAT AND CP COMMANDS
1. vi
2.cat
3. cp
Enter your choice: 1

----vi command------
Enter the file name: SUBA

PLEASE TERMINATE THE FILE USING ~
I AM SUBASRI
~


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



```
I AM SUBASRI
```

```
MENU OF OPERATIONS
C SIMULATION OF VI CAT AND CP COMMANDS
1. vi
2.cat
3. cp
Enter your choice: 3

----cp command-----
Enter the source file name: SUBA

Enter the destination file name: SRI

File is copied successfully

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

Result:

The program to implement the stimulation of vi, cat and cp commands was successful.

Ex No : 8	Simulation of Producer Consumer Problem
13.05.2024	

Aim:

To write a C++ program to implement the stimulation of Producer consumer problems.

Algorithm:

```
1.  Start the execution of the program.
2.  Declare the required variables.
3.  Get the ‘buffer count’ as user input.
4.  Use ‘switch statement’ to get choice from the user.
5.  Producer case:
        i. If (c_p==1 and count==1)
            PRINT ‘Consumer is ready’.
        ii. Else
            Buffer is full.
6.  Consumer case:
        i. If (p_p==1 and count==4)
            PRINT ‘Producer is ready’.
        ii. Else
            Buffer is empty.
7.  Buffer data:
        i.Call view function -> view
        ii.for: 1 to n.
        iii. PRINT Buffer data.
8.  Exit.
9.  Stop the execution of the program.
```

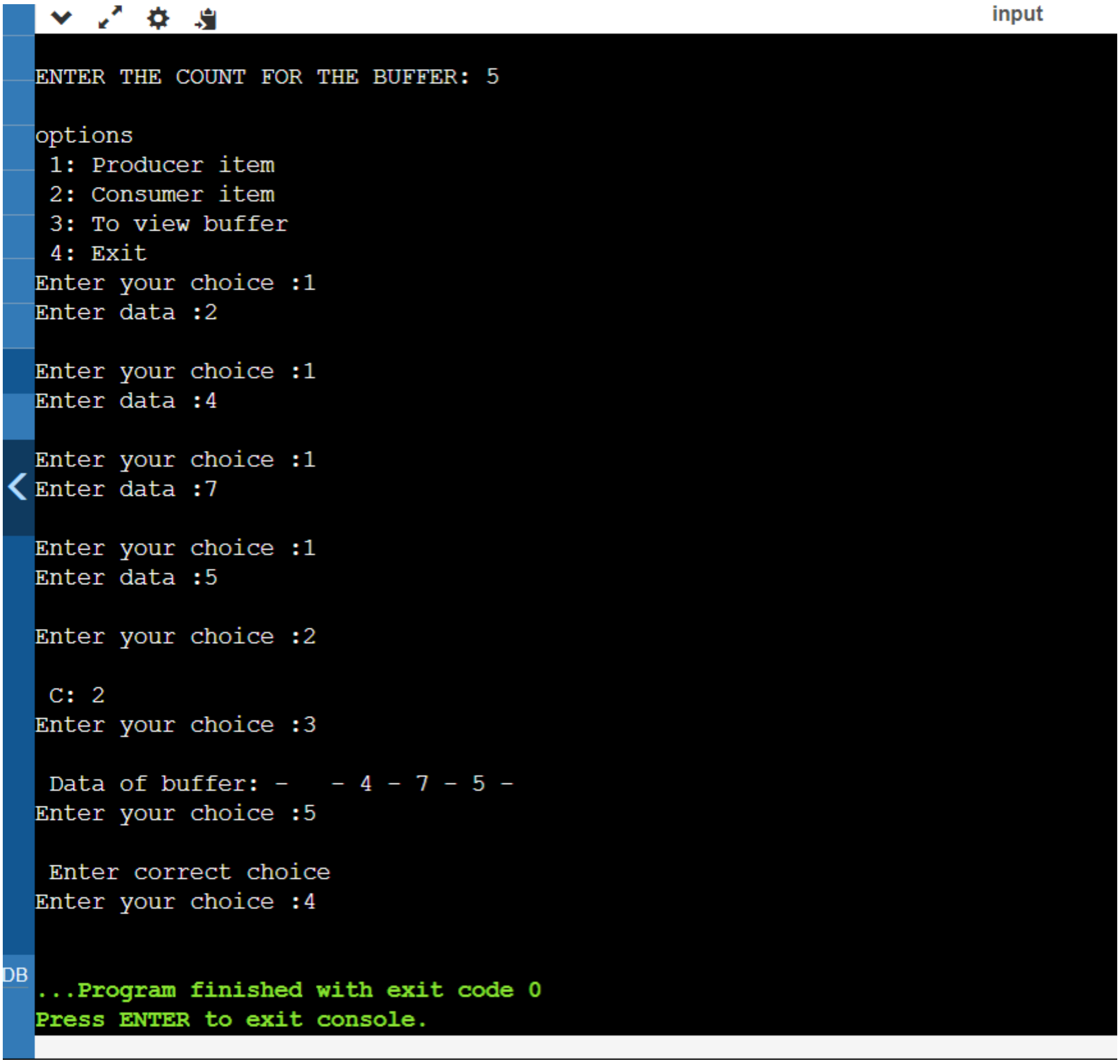
Program:

```
#include<stdio.h>
#include<conio.h>
int count=0;
int front=0;
int rear=0;
char buffer[25];
int p_p=0;
int c_p=0;
void producer(int);
void consumer(int);
void view(int);
void producer(int n)
{
char item;
if (count<n)
{
```

```
printf("Enter data :");
scanf(" %c",&item);
buffer [front]=item;
front = (front+1)%5;
count++;
if(c_p==1 && count==1)
{
printf("\n Consumer is now ready ");
}
}
else

{
printf("\n Buffer is full...");
p_p=1;
}
}
void consumer(int n)
{
char item;
if (count>0)
{
item = buffer[rear];
buffer[rear]=' ';
printf("\n C: %c",item);
rear=(rear+1)%5;
count--;
if(p_p==1 && count==4)
{
printf("\n Producer is now ready"); }
}
else
{
printf("\n Buffer is empty...");
c_p=1;
}
}
void view(int n)
{
int i;
printf("\n Data of buffer: ");

for(i=0;i<n;i++)
{
printf("- %c ",buffer[i]);
}
}
int main()
```

```
{
int i,n,ch,f=0;
printf("\nENTER THE COUNT FOR THE BUFFER: "); scanf("%d",&n);
printf("\noptions");
printf("\n 1: Producer item ");
printf("\n 2: Consumer item ");
printf("\n 3: To view buffer ");
printf("\n 4: Exit");
do
{
printf("\nEnter your choice :");
scanf("%d",&ch);
switch(ch)
{
case 1:producer(n);
break;
case 2:consumer(n);
break;
case 3:view(n);
break;
case 4:f=1;
break;
default:printf("\n Enter correct choice"); break;

}
}while(f==0);
}
```

Output:



```
ENTER THE COUNT FOR THE BUFFER: 5

options
 1: Producer item
 2: Consumer item
 3: To view buffer
 4: Exit
Enter your choice :1
Enter data :2

Enter your choice :1
Enter data :4

Enter your choice :1
Enter data :7

Enter your choice :1
Enter data :5

Enter your choice :2

 C: 2
Enter your choice :3

 Data of buffer: -   - 4 - 7 - 5 -
Enter your choice :5

 Enter correct choice
Enter your choice :4


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

Result:

The program has been successfully compiled and the stimulation of producer consumer problems is implemented.

Ex No : 9	Banker's Algorithm- Safety algorithm and Resource Request Algorithm
14.05.2024	

AIM:

To illustrate Banker's algorithm using c programming.

ALGORITHM:

1. Start.

2. Declare variables.

3.Create a for loop to get values.

4. Safety Method:
   In this method check whether the resources can be allocated or not and Ensure safety of process.

5. Resource Request Method:
   In this method check whether the additional request can be grant or not.

6. In Banker's method calculate need matrix
   Need= Maximum - Allocation

7. Input the values of Allocation,Maximum and available.

8.Display Allocation,Maximum and Need matrix.

9. Ask whether there is an resource request
   - If yes, then get the request.
   - Else, Display output.

10. Stop.

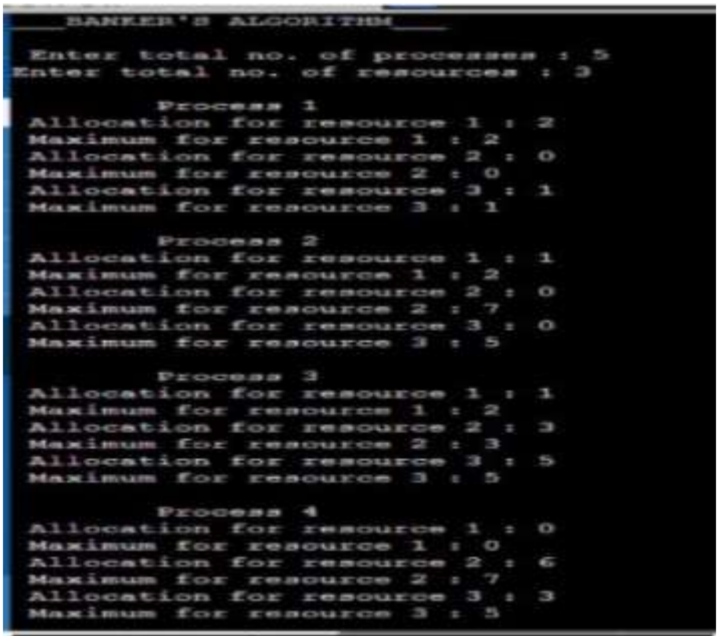
PROGRAM:

```
#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
```

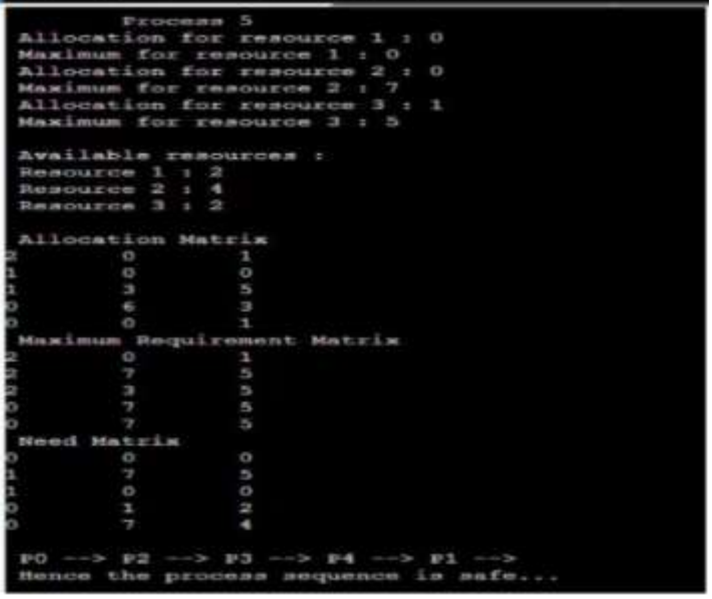
```
reqmat[1][10];
int i;
printf("\n___FOR ADDITINAL REQUEST:___");
printf("\n Enter additional request :- \n");
for(i=0;i<m;i++){ printf(" Request for
resource %d : ",i+1);
scanf("%d",&reqmat[0][i]);
} for(i=0;i<m;i++) if(reqmat[0][i] >
N[pid][i]){ printf("\n The request can be
granted.\n"); exit(0); } 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("Enter total no. of resources : ");
scanf("%d",m); for(i=0;i<*n;i++){
printf("\n\tProcess %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
Hence the process sequence is safe...\n"); return 1;
}else{ printf("\n The process sequence is not
safe...\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___BANKER'S ALGORITHM___\n");
accept(A,N,M,W,&n,&m); ret=banker(A,N,W,n,m); if(ret !=0
){ printf("\n 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;
}
```

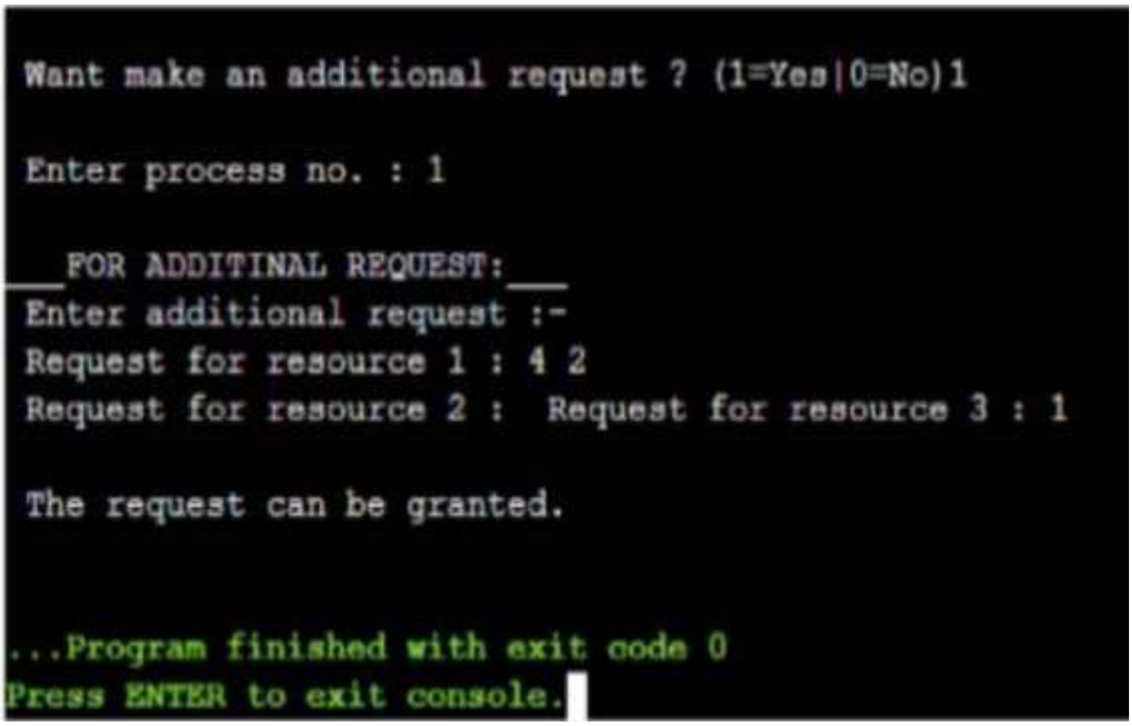
OUTPUT:



```
____BANKER'S ALGORITHM____

 Enter total no. of processes : 5
Enter total no. of resources : 3

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

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

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

        Process 4
 Allocation for resource 1 : 0
 Maximum for resource 1 : 0
 Allocation for resource 2 : 6
 Maximum for resource 2 : 7
 Allocation for resource 3 : 3
 Maximum for resource 3 : 5
```



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

 Available resources :
 Resource 1 : 2
 Resource 2 : 4
 Resource 3 : 2

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

 P0 --> P2 --> P3 --> P4 --> P1 -->
 Hence the process sequence is safe...
```



```
Want make an additional request ? (1=Yes|0=No)1

Enter process no. : 1

___FOR ADDITINAL REQUEST:___
Enter additional request :-
Request for resource 1 : 4 2
Request for resource 2 :  Request for resource 3 : 1

The request can be granted.


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

RESULT:

The Above Experiment Is Verified and Output came Successfully.

Ex No : 10	Dynamic Allocation Strategies
15.05.2024	

AIM:

To illustrate First fit, Best fit, Worst fit using c programming.

ALGORITHM:

1. Start.

2. Declare variables.

3.Input no. of blocks.

4. Using for loop get input according to allocated side.

```
5. Create loop to check in which memory block the remaining j
   Higher i memory block having higher remaining memory then assign
   Process to that memory block.
        b[j]    =   b[j]-F[i]
        Frogi =   Frogi + b[j]
```

6. If no then repeat 5th step for all process.

7.if space is insufficient then display wait.

8. Stop.

ALGORITHM(FIRST FIT):

1.Start.

2.Declare Variable.

3.Input memory block and processes with zero.

4.Start picking process and check where it assign.

5.If size of process <= size of block then assign, check for other process.

6.If space is not available, display wait.

7.Display allocated processers.

8.Stop.

ALGORITHM(BEST FIT):

1.Start.

2.Decalre variable.

3.Get direct to input process and memory block.

4.Using for loop, find max size that can be assign to current processes
 If it is found then assign it to the block.

5.If not, go and check for next process.

6.Check all process and Allocate in the block.

7.If there is size limitation display wait.

8.Display Allocation table.

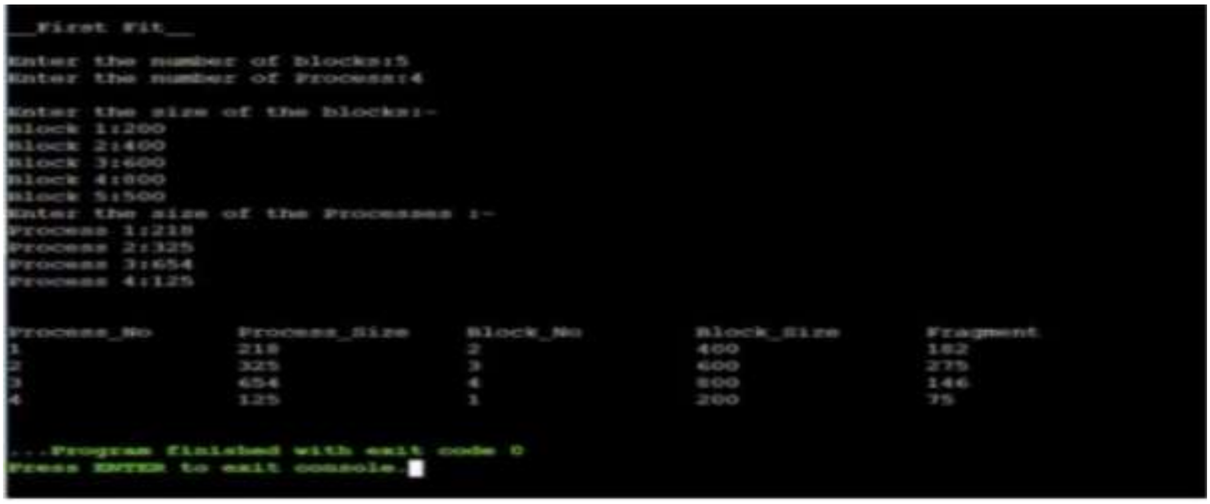
9.Stop.

PROGRAM:

FIRST FIT:

```
#include<stdio.h> #define max 25 void main(){ int
frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0; static int
bf[max],ff[max];int flag,flagn[max],fragi = 0,fragx =
0; printf("\n__First Fit__\n"); printf("\nEnter
the number of blocks:"); scanf("%d",&nb);
printf("Enter the number of Process:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++) { printf("Block %d:",i);
scanf("%d",&b[i]);
ff[i] = i;
}
printf("Enter the size of the Processes :-\n");
for(i=1;i<=nf;i++) { printf("Process %d:",i);
scanf("%d",&f[i]);
} int x =
1;
printf("\n\n
Process_
No\tProce
ss_Size\tBlock_No\tB
lock_Size\t
Fragment\
n
"); for(i=1;i<=nf;i++){ flag
= 1;
for(j=x;j<=nb;j++){
if(f[i] <= b[j]){ flagn[j]
= 1;
printf("%-15d\t%-15d\t%-15d\t%-15d\t",i, f[i],ff[j],b[j]); b[j]
= b[j] - f[i]; fragi = fragi + b[j];
printf("%-15d\n",b[j]); break;
}
else{ flagn[j] = 0;
x = 1; flag++;
} }
if(flag > nb) printf("%-15d\t%-15d\t%-15s\t%-15s\t%-
15s\n",i,f[i],"Has to wait...","...","..."); }}
```

OUTPUT:

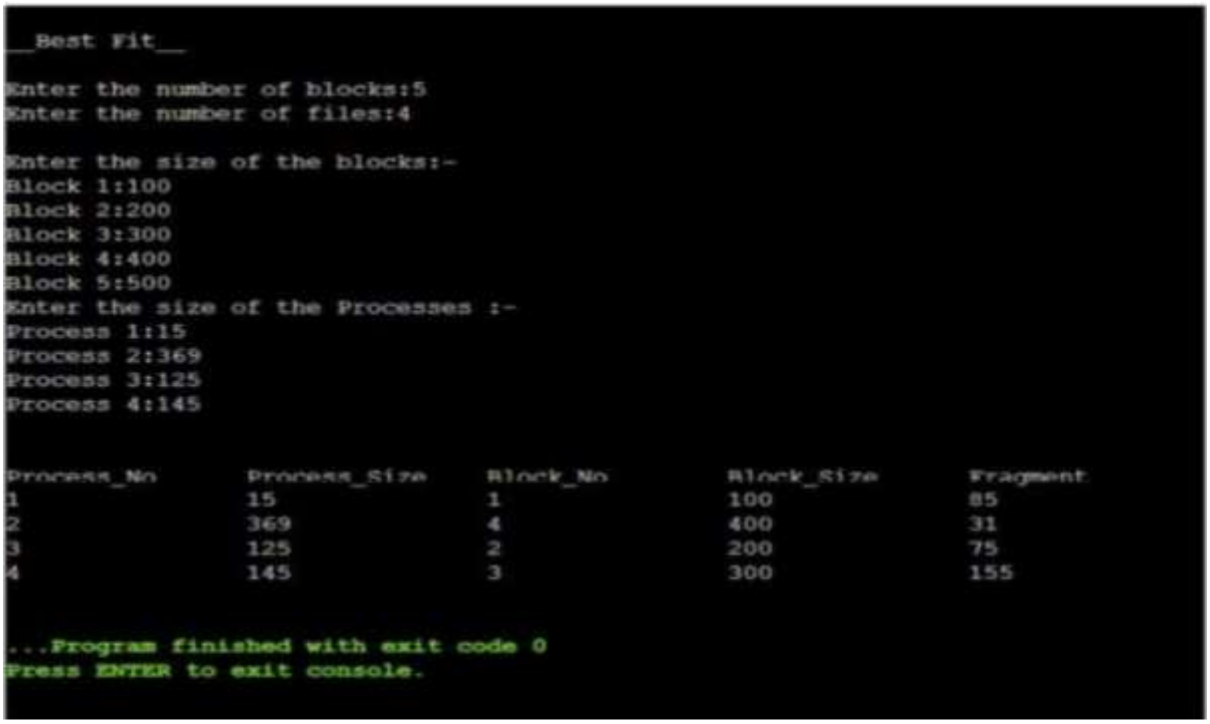


```
__First Fit__

Enter the number of blocks:5
Enter the number of Process:4

Enter the size of the blocks:-
Block 1:200
Block 2:400
Block 3:600
Block 4:800
Block 5:500
Enter the size of the Processes :-
Process 1:218
Process 2:325
Process 3:654
Process 4:125


Process_No      Process_Size    Block_No        Block_Size      Fragment
1               218             2               400             182
2               325             3               600             275
3               654             4               800             146
4               125             1               200             75


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

BEST FIT:

```
#include<stdio.h>
#define max 25 void
main()
{ int
frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000; st
atic int bf[max],ff[max],fragi = 0; printf("\n__Best
Fit__\n"); printf("\nEnter
the number of blocks:"); scanf("%d",&nb);
printf("Enter the
number of files:"); scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)
{
printf("Block %d:",i); scanf("%d",&b[i]);
ff[i] = i;
}
printf("Enter the size of the Processes :\n");
for(i=1;i<=nf;i++) { printf("Process
%d:",i); scanf("%d",&f[i]);
}
int y,m,z,temp1,flag; for(y=1;y<=nb;y++)
{
for(z=y;z<=nb;z++)
{
if(b[y]>b[z])
{
temp=b[y]; b[y]=b[z];
b[z]=temp;
temp1=ff[y]; ff[y]=ff[z];
ff[z]=temp1;
}
}
}
int flagn[max]; int
fragx = 0;
printf("\n\nProcess_No\tProcess_Size\tBlock_No\tBlock_Size\tFragment\n
"); for(i=1;i<=nf;i++)
{
flag = 1; for(j=1;j<=nb;j++){ if(f[i] <=
b[j]){ flagn[j]
= 1;
printf("%-15d\t%-15d\t%-15d\t%-15d\t",i, f[i],ff[j],b[j]); b[j]
= b[j] - f[i];
fragi = fragi + b[j]; printf("%-15d\n",b[j]);
break;
}
else {flagn[j]
= 0; flag++;
}
}
if(flag > nb) printf("%-15d\t%-15d\t%-15s\t%-15s\t%-
15s\n",i, f[i],"Has to wait..","...","..."); }}
```

OUTPUT:



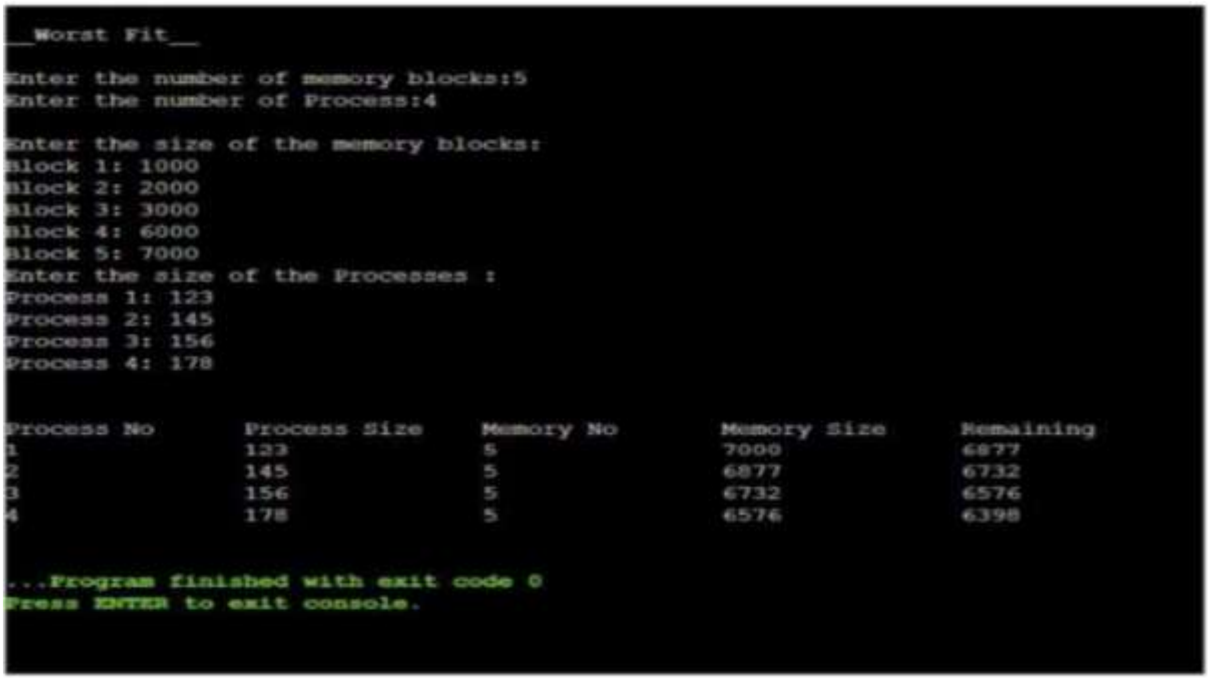
```
__Best Fit__

Enter the number of blocks:5
Enter the number of files:4

Enter the size of the blocks:-
Block 1:100
Block 2:200
Block 3:300
Block 4:400
Block 5:500
Enter the size of the Processes :-
Process 1:15
Process 2:369
Process 3:125
Process 4:145


Process_No      Process_Size    Block_No        Block_Size      Fragment
1               15              1               100             85
2               369             4               400             31
3               125             2               200             75
4               145             3               300             155


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

WORST FIT:

```
#include<stdio.h>
#define max 25 void
main()
{ int
frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0; sta
tic int bf[max],ff[max];int flag,fragi = 0;
printf("\n__Worst Fit__\n");
printf("\nEnter the number of memory
blocks:"); scanf("%d",&nb);
printf("Enter the number of Process:");
scanf("%d",&nf); printf("\nEnter the size
of the memory blocks:\n");
for(i=1;i<=nb;i++)
{
printf("Block %d: ",i); scanf("%d",&b[i]);
ff[i] = i;
}printf("Enter the size of the Processes
:\n"); for(i=1;i<=nf;i++)
{
printf("Process %d: ",i); scanf("%d",&f[i]);
}
int y,z,temp1; for(y=1;y<=nb;y++)
{
for(z=y;z<=nb;z++)
{
if(b[y]<b[z])
{
```

```
temp=b[y];
b[y]=b[z]; b[z]=temp;
temp1=ff[y];
ff[y]=ff[z];
ff[z]=temp1;
}
}
}
int flagn[max];
int fragx = 0;
printf("\n\nProcess No\tProcess Size\tMemory
No\tMemory Size\tRemaining\n");
for(i=1;i<=nf;i++)
{
flag = 1;
for(j=1;j<=nb;j++)
{
if(f[i] <= b[j])
{
flagn[j] = 1; printf("%-15d\t%-15d\t%-15d\t%-15d\t",i,
f[i],ff[j],b[j]); b[j] = b[j] - f[i]; fragi = fragi + b[j]; printf("%-
15d\n",b[j]); break;
}
else
{ flagn[j] =
0;
flag++;
}
}
if(flag > nb) printf("%-15d\t%-15d\t%-15s\t%-15s\t%-
15s\n",i,f[i],"Has to wait..","..","..");
}
}
```

OUTPUT:



```
__Worst Fit__

Enter the number of memory blocks:5
Enter the number of Process:4

Enter the size of the memory blocks:
Block 1: 1000
Block 2: 2000
Block 3: 3000
Block 4: 6000
Block 5: 7000
Enter the size of the Processes :
Process 1: 123
Process 2: 145
Process 3: 156
Process 4: 178


Process No      Process Size    Memory No       Memory Size     Remaining
1               123             5               7000            6877
2               145             5               6877            6732
3               156             5               6732            6576
4               178             5               6576            6398


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

RESULT:

The Above Experiment Is Verified and Output came Successfully.

Ex No : 11	**FIFO and Optimal page replacement algorithm**
18.05.2024	

AIM:

To illustrate FIFO and optimal page replacement algorithm using C.

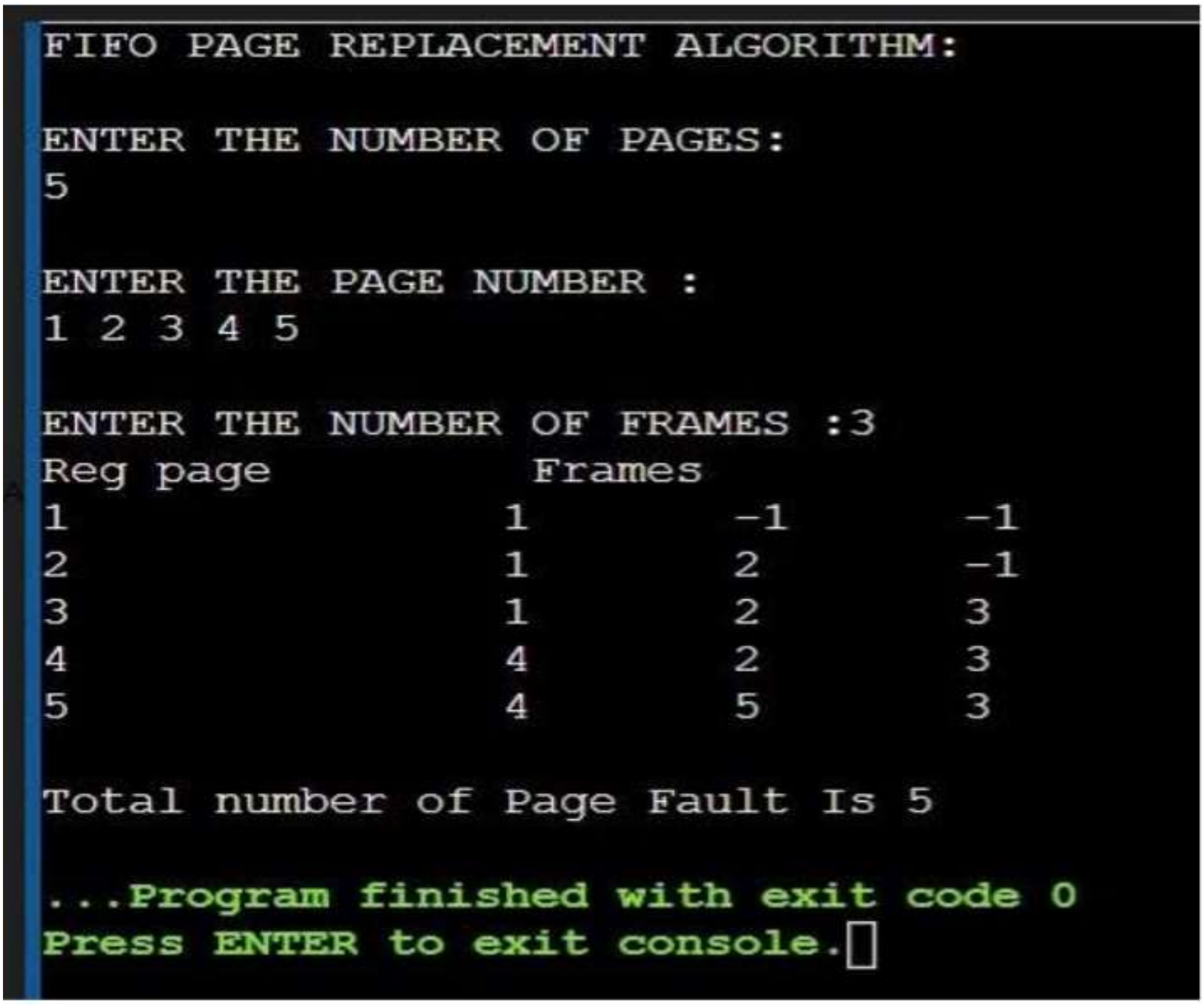
ALGORITHM:

```
  FIFO
    i) Start
    ii) Declare variable
    iii) input page number, frames from user
    iv) Check the need of replacement from pulled page to new page in memory using for
loop.
          1)  If frame [k]= a[i]
                Initial available=1
          2)  If (available =0)
                Assign frame[j]=[i]
                j=(j+1); 10
    v) Create a queue to hold the page.
    vi) get page w & insert in queue.
    vii) Check page fault.
    viii) Display page number r total number of page forced.
    ix) Stop.
```

CODE:

```
#include<stdio.h>
int main()
{
int
i,j,n,a[50],frame[10],no,k,avail,count=0; printf("__
_FIFO PAGE REPLACEMENT
ALGORITHM:___"); printf("\n\nENTER THE
NUMBER OF PAGES:\n"); scanf("%d",&n);
printf("\nENTER THE PAGE NUMBER :\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("\nENTER THE NUMBER OF
FRAMES :"); scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]= -1;
j=0;
printf("Reg page\t Frames\n");
for(i=1;i<=n;i++)
{
printf("%d\t\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if (avail==0)

{
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);
}
printf("\n");
}
printf("\nTotal number of Page Fault Is
%d",count); return 0;
}
```

OUTPUT:



```
FIFO PAGE REPLACEMENT ALGORITHM:

ENTER THE NUMBER OF PAGES:
5

ENTER THE PAGE NUMBER :
1 2 3 4 5

ENTER THE NUMBER OF FRAMES :3
Reg page          Frames
1                1       -1       -1
2                1       2        -1
3                1       2        3
4                4       2        3
5                4       5        3

Total number of Page Fault Is 5

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

ALGORITHM

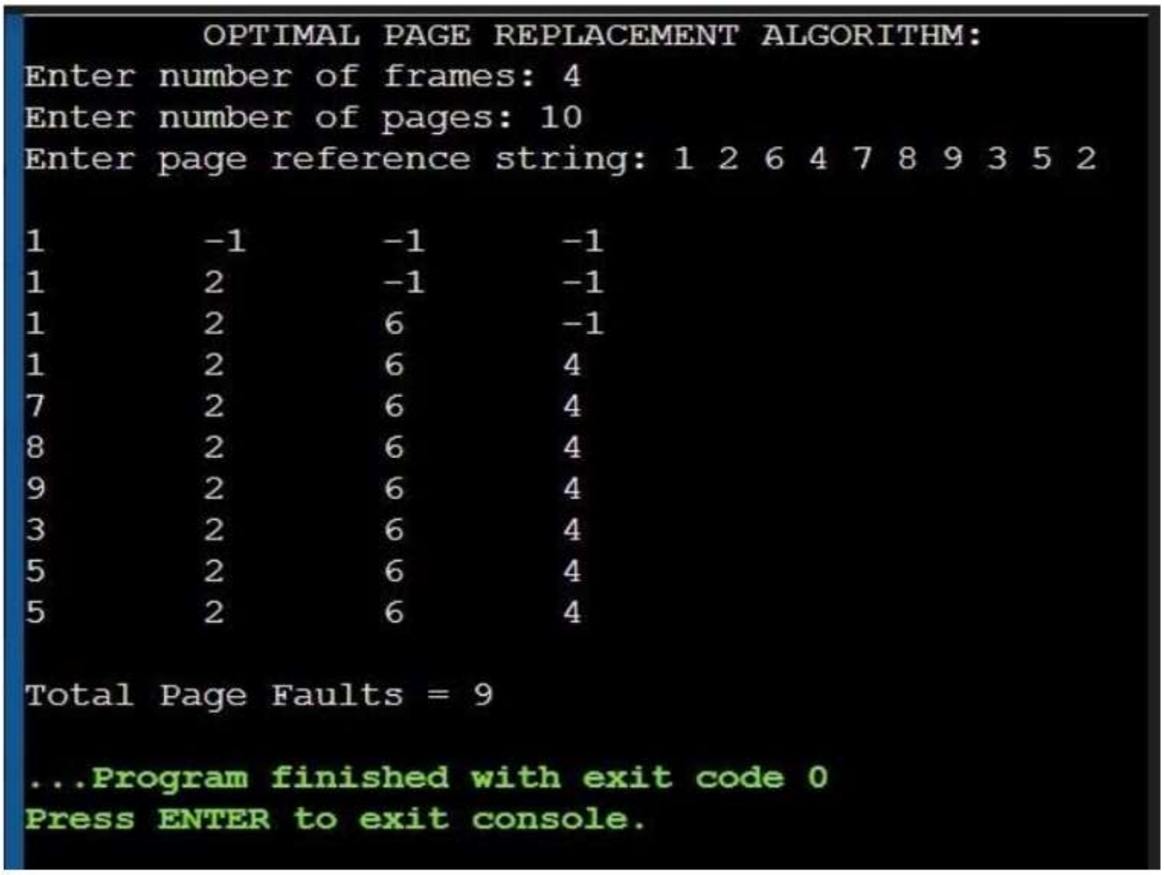
OPTIMAL:

```
1.Start.
2.Declare variable.
3.Get an input.
4.If a refused  is already present increment count.
5.If not available, Find a page that will not use longer period and Then replace the
page with new page.
6.Stop.
```

CODE:

```
#include<stdio.h>
int main()
{
int no_of_frames, no_of_pages, frames[10], pages[30], temp[10],
flag1, flag2, flag3, i, j, k,
pos, max, faults = 0;
printf("___OPTIMAL PAGE REPLACEMENT
ALGORITHM:___\n"); printf("Enter number of frames:
");
scanf("%d", &no_of_frames);
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("Enter page reference string: ");
for(i = 0; i < no_of_pages; ++i){
scanf("%d", &pages[i]);
}
for(i = 0; i < no_of_frames;
++i){ frames[i] = -1;
}
for(i = 0; i < no_of_pages;
++i){ flag1 = flag2 = 0;
for(j = 0; j < no_of_frames;
++j){ if(frames[j] ==
pages[i]){ flag1 = flag2 =
1;
break;
}
}

if(flag1 == 0){
for(j = 0; j < no_of_frames;
++j){ if(frames[j] == -1){
faults++;
frames[j] = pages[i];
flag2 = 1;
break;
}
}
}
if(flag2 == 0){
flag3 =0;
for(j = 0; j < no_of_frames; ++j){
temp[j] = -1;
for(k = i + 1; k < no_of_pages;
++k){ if(frames[j] ==
pages[k]){
temp[j] = k;
break
;
}}}
for(j = 0; j < no_of_frames;
++j){ if(temp[j] ==
-1){
pos = j;
flag3 = 1;
break; }}

if(flag3 ==0){
max = temp[0];
pos = 0;
for(j = 1; j < no_of_frames;
++j){ if(temp[j] > max){
max = temp[j];
pos = j; }}}
frames[pos] = pages[i];
faults++; }
printf("\n");
for(j = 0; j < no_of_frames;
++j){ printf("%d\t",
frames[j]);
}}
printf("\n\nTotal Page Faults = %d", faults);
return 0; }
```

OUTPUT:



```
         OPTIMAL PAGE REPLACEMENT ALGORITHM:
Enter number of frames: 4
Enter number of pages: 10
Enter page reference string: 1 2 6 4 7 8 9 3 5 2

1       -1      -1      -1
1       2       -1      -1
1       2       6       -1
1       2       6       4
7       2       6       4
8       2       6       4
9       2       6       4
3       2       6       4
5       2       6       4
5       2       6       4

Total Page Faults = 9

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

RESULT:

The above code is verified and output came successfully.