NAME: Krish Jain
REG NO: 21BCE1598

EXP1

Basic Unix commands

- General commands

1) **cal <month> <year>** : Gives the calender of the mentioned month and year

```
mint@mint:~$ cal 05 2023
      May 2023
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

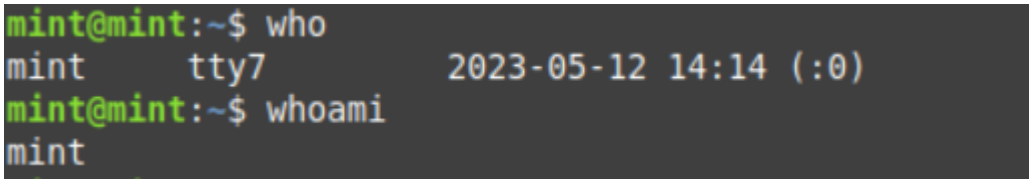
```
mint@mint:~$ cal 2023
                                2023
      January               February              March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4            1  2  3  4
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   5  6  7  8  9 10 11
15 16 17 18 19 20 21  12 13 14 15 16 17 18  12 13 14 15 16 17 18
22 23 24 25 26 27 28  19 20 21 22 23 24 25  19 20 21 22 23 24 25
29 30 31              26 27 28              26 27 28 29 30 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      1  2  3  4  5  6                1  2  3
 2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
30

        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         1  2  3  4  5                1  2
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
16 17 18 19 20 21 22  20 21 22 23 24 25 26  17 18 19 20 21 22 23
23 24 25 26 27 28 29  27 28 29 30 31        24 25 26 27 28 29 30
30 31

      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  3  4  5  6  7            1  2  3  4                1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
```

2) **man <command>:** Gives the description of the mentioned command

```
CAL(1)                          BSD General Commands Manual                          CAL(1)

NAME
     cal, ncal – displays a calendar and the date of Easter

SYNOPSIS
     cal [-3hjy] [-A number] [-B number] [[month] year]
     cal [-3hj] [-A number] [-B number] -m month [year]
     ncal [-3bhjJpwySM] [-A number] [-B number] [-W number] [-s country_code] [[month] year]
     ncal [-Jeo] [-A number] [-B number] [year]
     ncal [-CN] [-H yyyy-mm-dd] [-d yyyy-mm]

DESCRIPTION
     The cal utility displays a simple calendar in traditional format and ncal offers an alternative layout, more options
     and the date of Easter.  The new format is a little cramped but it makes a year fit on a 25x80 terminal.  If argu-
     ments are not specified, the current month is displayed.

     The options are as follows:

     -h      Turns off highlighting of today.

     -J      Display Julian Calendar, if combined with the -o option, display date of Orthodox Easter according to the Ju-
             lian Calendar.

     -e      Display date of Easter (for western churches).

     -j      Display Julian days (days one-based, numbered from January 1).

     -m month
             Display the specified month.  If month is specified as a decimal number, appending 'f' or 'p' displays the
             same month of the following or previous year respectively.

     -o      Display date of Orthodox Easter (Greek and Russian Orthodox Churches).

     -p      Print the country codes and switching days from Julian to Gregorian Calendar as they are assumed by ncal.
             The country code as determined from the local environment is marked with an asterisk.

     -s country_code
Manual page cal(1) line 1 (press h for help or q to quit)
```

3) **Who / whoami ::** who command is a tool print information about users who are currently logged in.

   whoami : This command gives details about the current user.



4) **Clear :** This command clears the terminal screen

5) **Find:** The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them.

```
mint@mint:~$ find
.
./.lesshst
./os
./.cinnamon
./.cinnamon/configs
./.cinnamon/configs/menu@cinnamon.org
./.cinnamon/configs/menu@cinnamon.org/0.json
./.cinnamon/configs/network@cinnamon.org
./.cinnamon/configs/network@cinnamon.org/network@cinnamon.org.json
./.cinnamon/configs/sound@cinnamon.org
./.cinnamon/configs/sound@cinnamon.org/sound@cinnamon.org.json
./.cinnamon/configs/calendar@cinnamon.org
./.cinnamon/configs/calendar@cinnamon.org/13.json
./.cinnamon/configs/power@cinnamon.org
./.cinnamon/configs/power@cinnamon.org/power@cinnamon.org.json
./.cinnamon/configs/grouped-window-list@cinnamon.org
./.cinnamon/configs/grouped-window-list@cinnamon.org/2.json
./.cinnamon/configs/favorites@cinnamon.org
./.cinnamon/configs/favorites@cinnamon.org/9.json
./.cinnamon/configs/notifications@cinnamon.org
./.cinnamon/configs/notifications@cinnamon.org/notifications@cinnamon.org.json
./.cinnamon/configs/show-desktop@cinnamon.org
./.cinnamon/configs/show-desktop@cinnamon.org/1.json
./.cinnamon/configs/printers@cinnamon.org
./.cinnamon/configs/printers@cinnamon.org/6.json
./Videos
./Pictures
./Music
./Documents
./Documents/os lab
./Public
./Templates
./Downloads
./.xsession-errors
./.Xauthority
./.local
./.local/share
./.local/share/recently-used.xbel
```

6) **Id:** id command in Linux is used to find out user and group names and numeric ID's (UID or group ID) of the current user or any other user in the server.

```
mint@mint:~$ id
uid=999(mint) gid=999(mint) groups=999(mint),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),115(lpadmin),135(sambashare)
```

7) **Echo:** This command is used as a print statement

```
mint@mint:~$ echo hello world 21BCE1598
hello world 21BCE1598
```

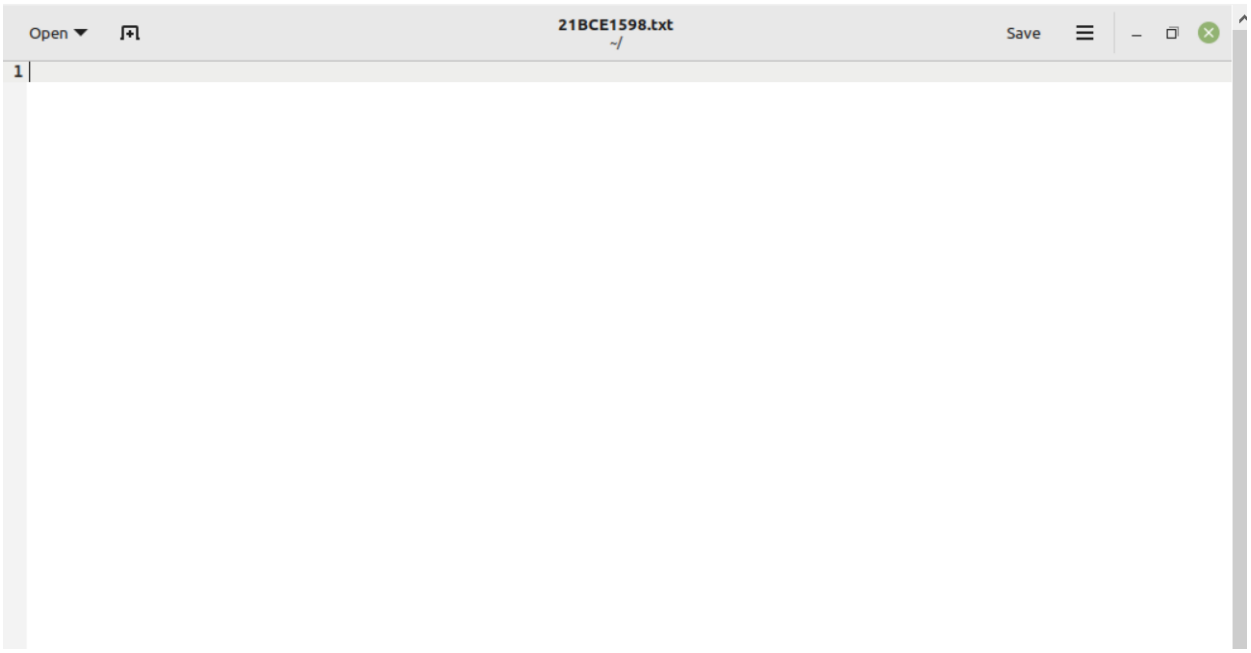
8) **Date:** Gives the current date and time of the system

```
mint@mint:~$ date
Fri May 12 14:53:11 UTC 2023
```

- **File manipulation commands :**

1) **Gedit <filename>:** To create and open a file

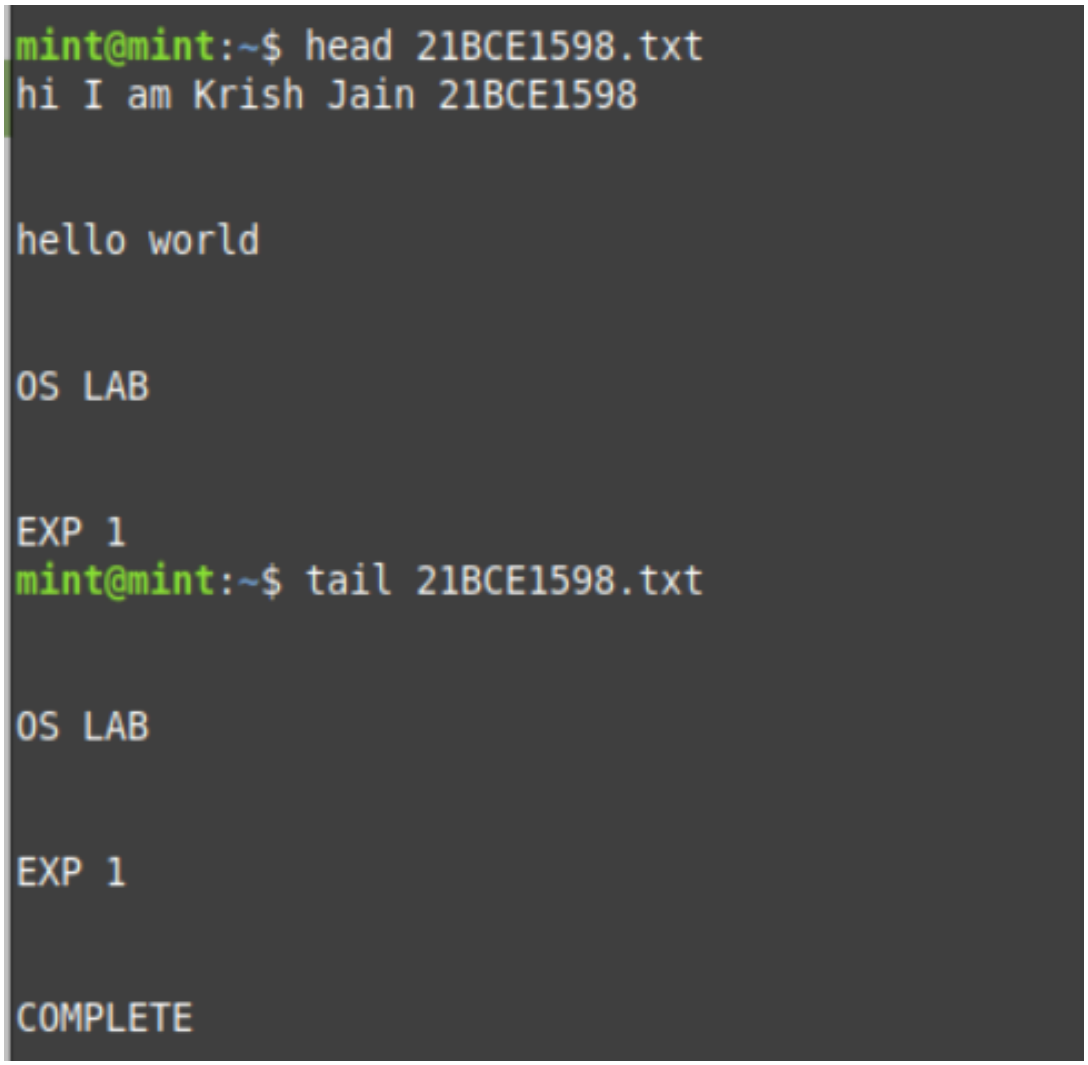
```
mint@mint:~$ gedit 21BCE1598.txt
mint@mint:~$ ls
 21BCE1598.txt   Desktop   Documents   Downloads   Music   Pictures   Public   T
emplates   Videos   'os '
mint@mint:~$
```



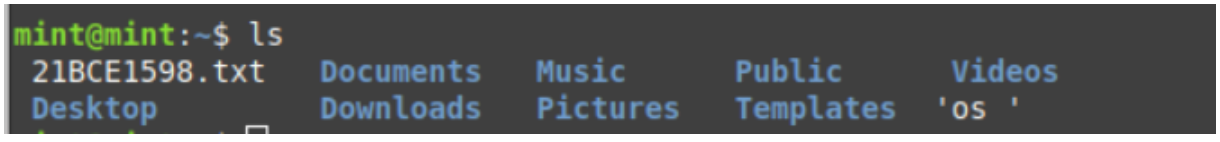
2) **Cat <filename> :** To display content of a file

```
mint@mint:~$ cat 21BCE1598.txt
hi I am Krish Jain 21BCE1598


hello world


OS LAB


EXP 1


COMPLETE
```

3) **Head <filename> / Tail <filename> :** Gives the first 10 lines of the file / Displays the last 10 lines of the file.



4) **ls :** This command lists out all the files in the current directory.

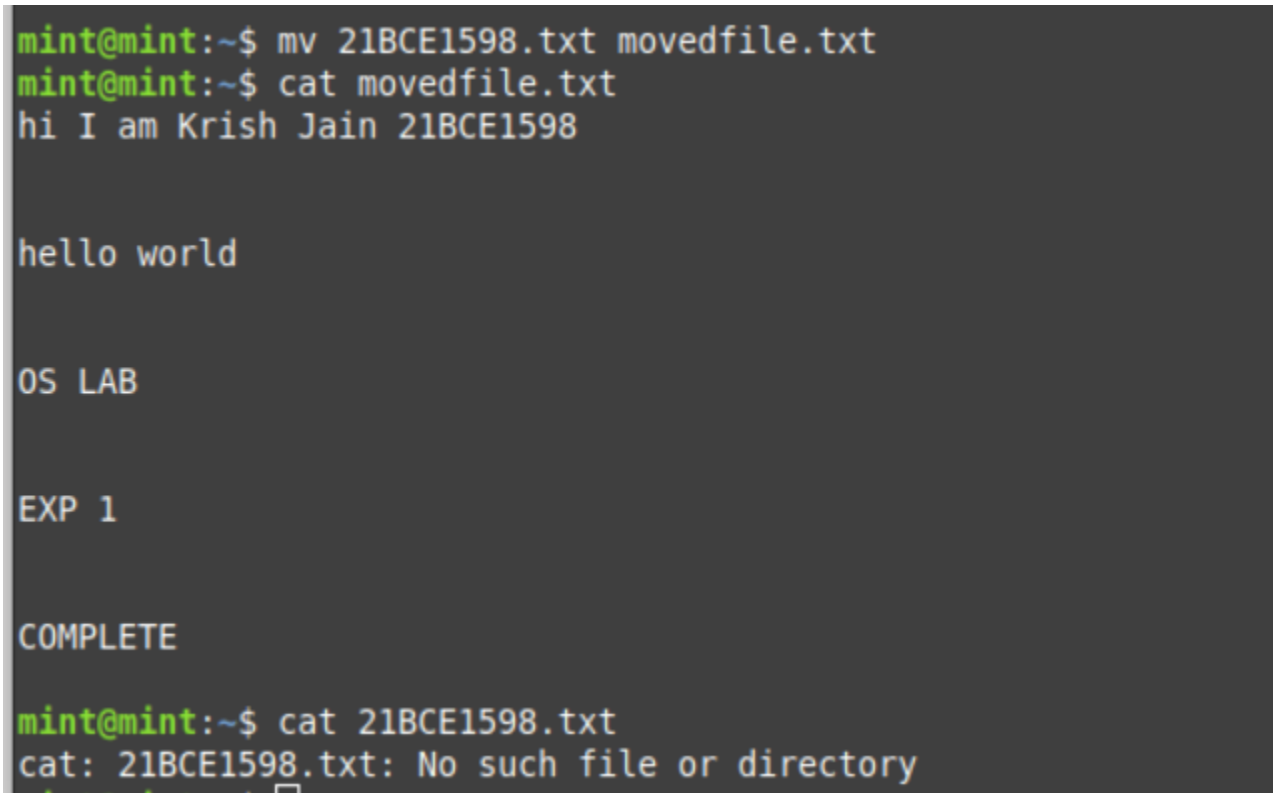


5) **Cp <source_file> <destination_file> :** Copy content from source file to destination.

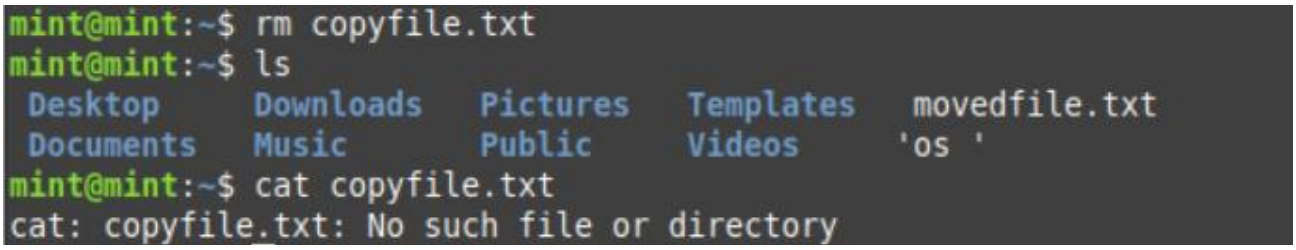
```
mint@mint:~$ cp 21BCE1598.txt copyfile.txt
mint@mint:~$ cat 21BCE1598.txt
hi I am Krish Jain 21BCE1598


hello world


OS LAB


EXP 1


COMPLETE
```

```
mint@mint:~$ cat copyfile.txt
hi I am Krish Jain 21BCE1598


hello world


OS LAB


EXP 1


COMPLETE
```

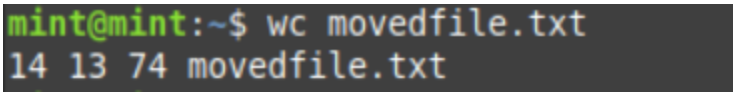
6) **mv <source_file> <destination_file> :** This command is used to move source file to the destination file mentioned in the command, the source file is removed after this command.



7) **rm** <file_name>: This command is used to to remove the file mentioned.



8) **wc <**file_name**>:** This command gives the word count of the file mentioned. It return 4 columns, first column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.



9) **cmp** <file_1> <file_2> : This command compares the 2 files mentioned and when cmp is used for comparison between two files, it reports the location of the first mismatch to the screen if difference is found and if no difference is found i.e the files compared are identical

```
mint@mint:~$ cmp movedfile.txt 21BCE1598.txt
movedfile.txt 21BCE1598.txt differ: byte 38, line 4
```

10) **diff** <file_1> <file_2> : This command is used to display the differences in the files by comparing the files line by line. Unlike its fellow members, 'cmp' and 'comm', it tells us which lines in one file have is to be changed to make the two files identical.

```
mint@mint:~$ diff movedfile.txt 21BCE1598.txt
4c4
< hello world
---
> hello python
```

11) **mkdir <dir_name> :** This is used to create a new directory.

```
mint@mint:~$ mkdir NewFolder
mint@mint:~$ ls
 21BCE1598.txt   Documents   Music       Pictures   Templates   movedfile.txt
 Desktop         Downloads   NewFolder   Public     Videos     'os '
mint@mint:~$
```

12) **rmdir <dir_name>:** rmdir stands for remove directory i.e. it will delete the mentioned directory.

```
mint@mint:~$ rmdir NewFolder
mint@mint:~$ ls
 21BCE1598.txt   Documents   Music      Public      Videos          'os '
 Desktop         Downloads   Pictures   Templates   movedfile.txt
mint@mint:~$
```

13) **cd <dir_name> :** It stands for change directory, used to change from one directory to the other.

```
mint@mint:~$ cd Desktop/
mint@mint:~/Desktop$
```

NAME: Krish Jain
REG NO: 21BCE1598

EXP2

Shell Programming

1) Write a shell program to Add two numbers

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p1_21BCE1598.sh
read -p "Enter first number: " num1
read -p "Enter second number: " num2
sum=$(( $num1 + $num2 ))
echo "Sum is: $sum"
```

```
mint@mint:~$ sh p1_21BCE1598.sh
Enter first number: 21
Enter second number: 23
Sum is: 44
```

2) Write a shell program to find largest number out of 3 numbers

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p2_21BCE1598.sh *
echo "Enter Num1: "
read num1
echo "Enter Num2: "
read num2
echo "Enter Num3: "
read num3
echo "Greatest Num :"
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
echo $num2
else
echo $num3
fi
```

```
mint@mint:~$ sh p2_21BCE1598.sh
Enter Num1:
21
Enter Num2:
2
Enter Num3:
43
Greatest Num :
43
```

3) Write a shell program for UNIX commands using case

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p3_21BCE1598.sh *
echo "Enter Your choice: "
echo "1. Clear"
echo "2. ls"
echo "3. pwd"
read command
case $command in
"2")
pwd;;
"3")
ls;;
"1")
clear;;
*)
echo "Invalid command";;
esac
```

```
mint@mint:~$ nano p3_21BCE1598.sh
mint@mint:~$ sh p3_21BCE1598.sh
Enter Your choice:
1. Clear
2. ls
3. pwd
```

4) Write a shell program to find factorial of a given number

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p4_21BCE1598.sh *
echo "Enter Number:"
read num
factorial=1
i=1
while [ $i -le $num ]
do
factorial=`expr $factorial \* $i`
i=`expr $i + 1`
done
echo "Factorial of $num: $factorial"
```

```
mint@mint:~$ nano p4_21BCE1598.sh
mint@mint:~$ sh p4_21BCE1598.sh
Enter Number:
5
Factorial of 5: 120
```

5) Write a shell program get fibonacci series upto given no. of terms

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p5_21BCE1598.sh *
echo "Enter number:"
read num_terms
a=0
b=1
i=1
echo "Fibonacci sequence: $a $b"
while [ $i -le $num_terms ]
do
c=$((a+b))
echo "$c"
a=$b
b=$c
i=`expr $i + 1`
done
```

```
mint@mint:~$ nano p5_21BCE1598.sh
mint@mint:~$ sh p5_21BCE1598.sh
Enter number:
5
Fibonacci sequence: 0 1
1
2
3
5
8
```

6) Write a shell script to get largest digit of a number

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p6_21BCE1598.sh *
echo "Enter number:"
read number
max_digit=`echo $number | cut -c1`
for i in `seq 2 ${#number}`
do
current_digit=`echo $number | cut -c$i`
if [ $current_digit -gt $max_digit ]
then
max_digit=$current_digit
fi
done
echo "The largest digit in $number is $max_digit."
```

```
mint@mint:~$ nano p6_21BCE1598.sh
mint@mint:~$ sh p6_21BCE1598.sh
Enter number:
4356
The largest digit in 4356 is 6.
```

7) Write a shell script to reverse a number

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p7_21BCE1598.sh *
echo "Enter number:"
read number
reversed_number=""
while [ $number -gt 0 ]
do
digit=`expr $number % 10`
reversed_number="$reversed_number$digit"
number=`expr $number / 10`
done
echo "The reversed number is $reversed_number."
```

```
mint@mint:~$ nano p7_21BCE1598.sh
mint@mint:~$ sh p7_21BCE1598.sh
Enter number:
2345
The reversed number is 5432.
```

8) To write a shell script to display student grade details

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                    p8_21BCE1598.sh *
echo "Enter student name:"
read name
echo "Enter marks obtained in English:"
read english_marks
echo "Enter marks obtained in Math:"
read math_marks
echo "Enter marks obtained in Science:"
read science_marks
total_marks=`expr $english_marks + $math_marks +
$science_marks`
average_marks=`expr $total_marks / 3`
if [ $average_marks -ge 90 ]
then
grade="A+"
elif [ $average_marks -ge 80 ]
then
grade="A"
elif [ $average_marks -ge 70 ]
then
grade="B"
```

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                        p8_21BCE1598.sh
elif [ $average_marks -ge 80 ]
then
grade="A"
elif [ $average_marks -ge 70 ]
then
grade="B"
elif [ $average_marks -ge 60 ]
then
grade="C"
else
grade="Fail"
fi
echo "Name: $name"
echo "English: $english_marks"
echo "Math: $math_marks"
echo "Science: $science_marks"
echo "Total: $total_marks"
echo "Average marks: $average_marks"
echo "Grade: $grade"
```

```
mint@mint:~$ nano p8_21BCE1598.sh
mint@mint:~$ sh p8_21BCE1598.sh
Enter student name:
krish
Enter marks obtained in English:
89
Enter marks obtained in Math:
95
Enter marks obtained in Science:
92
```

```
Name: krish
English: 89
Math: 95
Science: 92
```

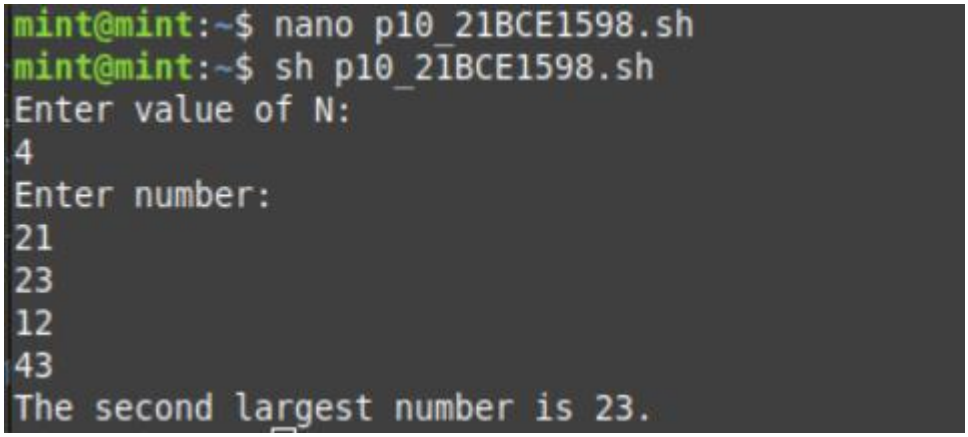
9) Write a shell script to get sum of N numbers

```
File  Edit  View  Search  Terminal  Help
 GNU nano 6.2                    p9_21BCE1598.sh *
echo "Enter value of N:"
read n
sum=0
echo "Enter numbers:"
for i in `seq 1 $n`
do
read temp
sum=`expr $sum + $temp`
done
echo "Sum of the given numbers is $sum."
```

```
mint@mint:~$ nano p9_21BCE1598.sh
mint@mint:~$ sh p9_21BCE1598.sh
Enter value of N:
3
Enter numbers:
21
23
21
Sum of the given numbers is 65.
```

10) To write a shell script to find the second largest number.

```
File  Edit  View  Search  Terminal  Help
 GNU nano 6.2                    p10_21BCE1598.sh *
n=0
echo "Enter value of N:"
read n
n=`expr $n - 1`
echo "Enter number:"
read largest
second_largest=0
for i in `seq 1 $n`
do
read num
if [ $num -gt $largest ]
then
second_largest=$largest
largest=$num
elif [ $num -gt $second_largest ]
then
second_largest=$num
fi
done
echo "The second largest number is $second_largest."
```


mint@mint:~$ nano p10_21BCE1598.sh
mint@mint:~$ sh p10_21BCE1598.sh
Enter value of N:
4
Enter number:
21
23
12
43
The second largest number is 23.

NAME: Krish Jain

REG NO: 21BCE1598

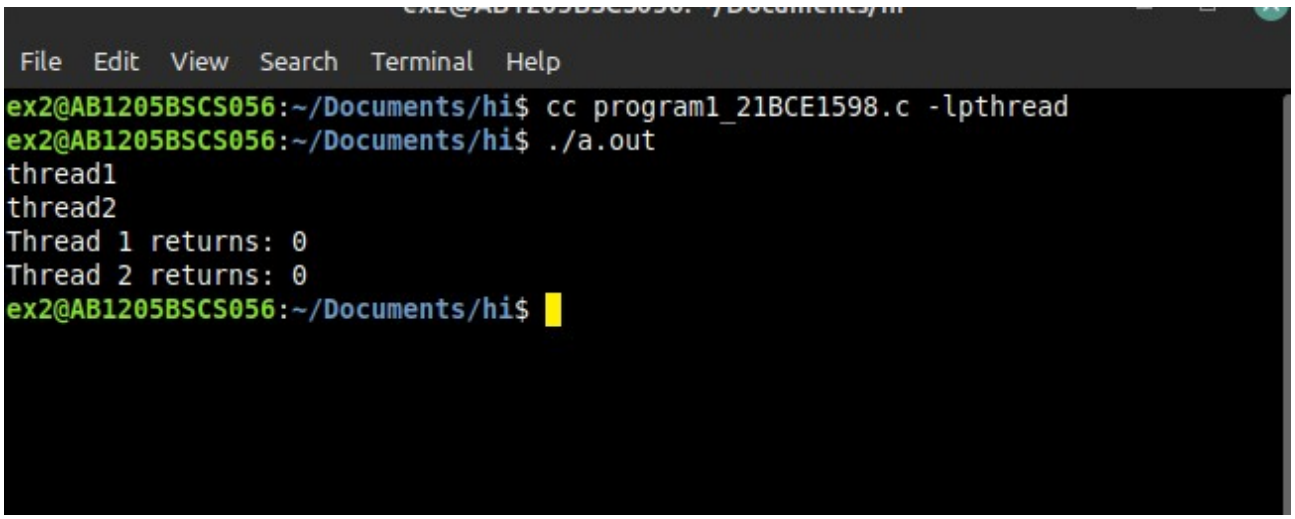
EXP 4

1) Create two threads thread1 and thread2 and call functions fun1 anf fun2 respectively.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void *print_message(void *ptr){
   char *message;
   message=(char*)ptr;
   printf("%s\n",message);
}

int main(){
   pthread_t thread1,thread2;
   char *message1="thread1";
   char *message2="thread2";
   int ir1,ir2;
   ir1=pthread_create(&thread1,NULL,print_message,(void *)message1);
   ir2=pthread_create(&thread2,NULL,print_message,(void *)message2);
   pthread_join(thread1,NULL);
   pthread_join(thread2,NULL);
   printf("Thread 1 returns: %d\n",ir1);
   printf("Thread 2 returns: %d\n",ir2);
   exit(0);
}
```

OUTPUT:

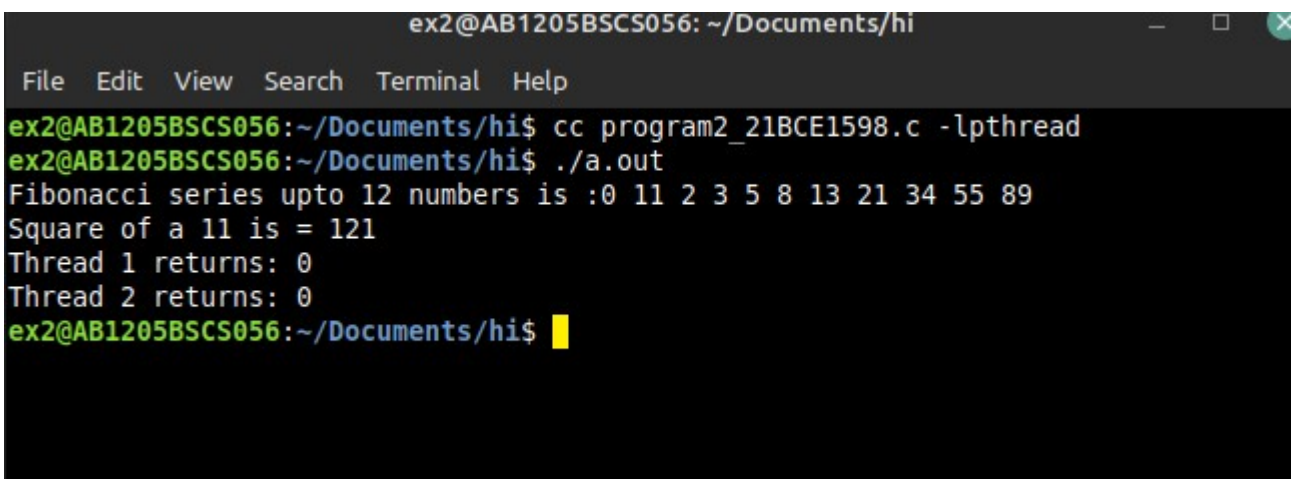


2) Create two threads thread1 and thread2 and call functions fun1 anf fun2 respectively.
Compute and print Finbonacci in fun1 and square of a number in fun2.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void *fun1(void *ptr){
   int value = *((int*)ptr),n1=0,n2=1,n3;
   printf("Fibonacci series upto %d numbers is :", value);
   printf("%d %d",0,1);
   for(int i=2;i<value; ++i) {
   n3=n1+n2;
   printf("%d ",n3);
   n1=n2;
   n2=n3;
   }
}
void *fun2(void *ptr) {
   int value = *((int *)ptr),sq;
   sq=value*value;
   printf("\nSquare of a %d is = %d\n", value, sq);
}

int main(){
   pthread_t thread1,thread2;
   int n1=12, n2=11;
   int ir1,ir2;
   ir1=pthread_create(&thread1,NULL,fun1,(void*)&n1);
   ir2=pthread_create(&thread2,NULL,fun2,(void*)&n2);
   pthread_join(thread1,NULL);
   pthread_join(thread2,NULL);
   printf("Thread 1 returns: %d\n",ir1);
   printf("Thread 2 returns: %d\n",ir2);
   exit(0);
}
```

OUTPUT:

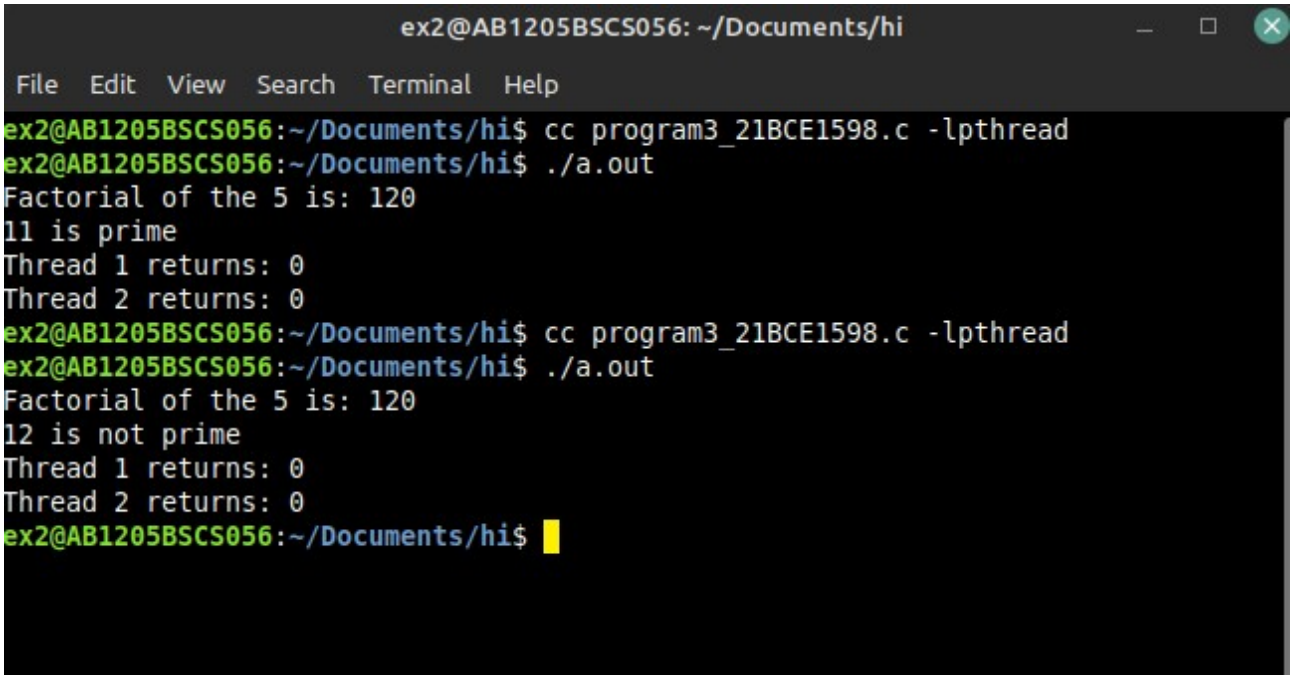


3) Create two threads thread1 and thread2 and call functions fun1 anf fun2 respectively. Compute and print Factorial in fun1 and Prime number in fun2.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void *fun1(void *ptr){
   int value = *((int*)ptr),factorial=1;
   for(int i=1;i<=value; i++) {
   factorial=factorial*i;
   }
   printf("Factorial of the %d is: %d\n",value,factorial);
}

void *fun2(void *ptr) {
   int n = *((int *)ptr),m=0,flag=0,i;
   m=n/2;
   for(i=2;i<=m;i++)
   {
   if(n%i==0)
   {
   printf("%d is not prime\n",n);
   flag=1;
   break;
   }
   }
   if(flag==0)
   printf("%d is prime\n",n);
}

int main(){
   pthread_t thread1,thread2;
   int n1=5, n2=12;
   int ir1,ir2;
   ir1=pthread_create(&thread1,NULL,fun1,(void*)&n1);
   ir2=pthread_create(&thread2,NULL,fun2,(void*)&n2);
   pthread_join(thread1,NULL);
   pthread_join(thread2,NULL);
   printf("Thread 1 returns: %d\n",ir1);
   printf("Thread 2 returns: %d\n",ir2);
   exit(0);


}
```

OUTPUT:

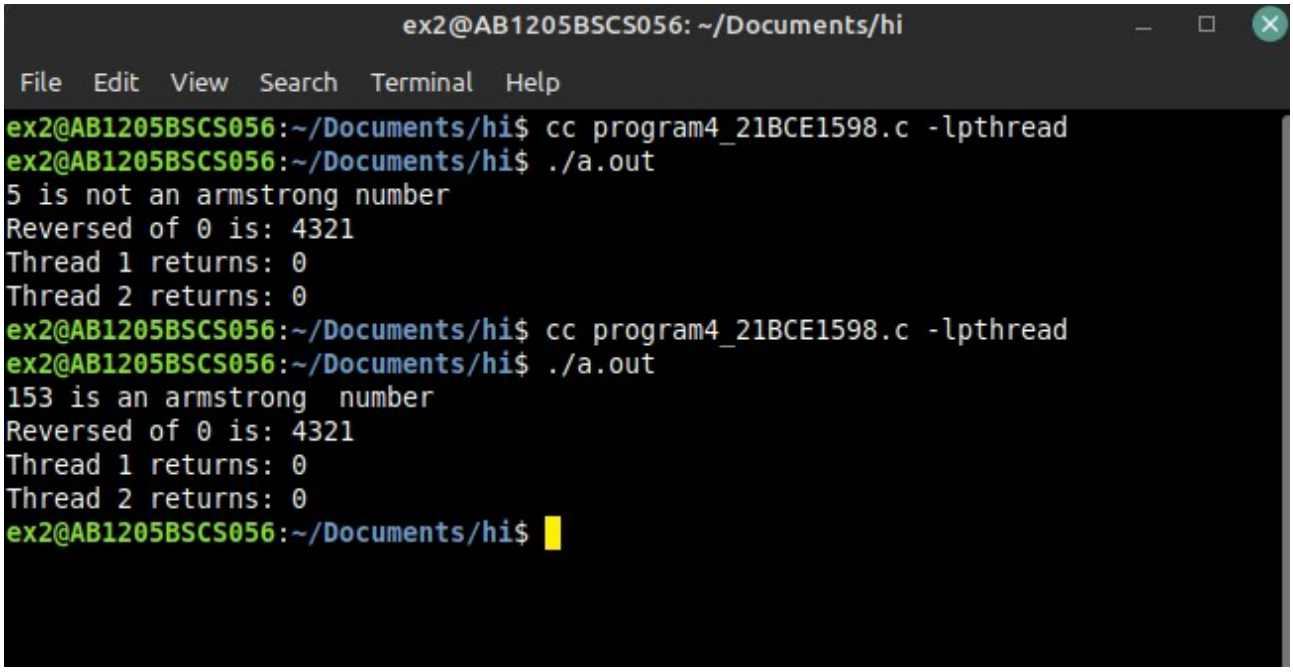

ex2@AB1205BSCS056: ~/Documents/hi
File Edit View Search Terminal Help
ex2@AB1205BSCS056:~/Documents/hi$ cc program3_21BCE1598.c -lpthread
ex2@AB1205BSCS056:~/Documents/hi$ ./a.out
Factorial of the 5 is: 120
11 is prime
Thread 1 returns: 0
Thread 2 returns: 0
ex2@AB1205BSCS056:~/Documents/hi$ cc program3_21BCE1598.c -lpthread
ex2@AB1205BSCS056:~/Documents/hi$ ./a.out
Factorial of the 5 is: 120
12 is not prime
Thread 1 returns: 0
Thread 2 returns: 0
ex2@AB1205BSCS056:~/Documents/hi$

4) Create two threads thread1 and thread2 and call functions fun1 anf fun2 respectively.
Compute and print Armstrong number or not in fun1 and Reverse number in fun2.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>

void *fun1(void *ptr){
   int n = *((int*)ptr);
   int k=n;
   int r,sum=0,temp;
   temp=n;
   while(n>0)
   {
   r=n%10;
   sum=sum+(r*r*r);
   n=n/10;
   }
   if(temp==sum)
   printf("%d is an armstrong  number\n",k);
   else
   printf("%d is not an armstrong number\n",k);
}

void *fun2(void *ptr) {
   int n = *((int *)ptr);
   int reverse=0, rem;
   while(n!=0)
   {
     rem=n%10;
     reverse=reverse*10+rem;
     n/=10;
   }
   printf("Reversed of %d is: %d\n",n,reverse);
}


int main(){
   pthread_t thread1,thread2;
   int n1=153, n2=1234;
   int ir1,ir2;
   ir1=pthread_create(&thread1,NULL,fun1,(void*)&n1);
   ir2=pthread_create(&thread2,NULL,fun2,(void*)&n2);
   pthread_join(thread1,NULL);
   pthread_join(thread2,NULL);
   printf("Thread 1 returns: %d\n",ir1);
   printf("Thread 2 returns: %d\n",ir2);
   exit(0);

}
```

OUTPUT:


ex2@AB1205BSCS056: ~/Documents/hi
File Edit View Search Terminal Help
ex2@AB1205BSCS056:~/Documents/hi$ cc program4_21BCE1598.c -lpthread
ex2@AB1205BSCS056:~/Documents/hi$ ./a.out
5 is not an armstrong number
Reversed of 0 is: 4321
Thread 1 returns: 0
Thread 2 returns: 0
ex2@AB1205BSCS056:~/Documents/hi$ cc program4_21BCE1598.c -lpthread
ex2@AB1205BSCS056:~/Documents/hi$ ./a.out
153 is an armstrong number
Reversed of 0 is: 4321
Thread 1 returns: 0
Thread 2 returns: 0
ex2@AB1205BSCS056:~/Documents/hi$

NAME: Krish Jain
REG NO: 21BCE1598

EXP 5

1) **Create a process and Parent ID and Child ID.**

CODE:

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

int main() {
pid_t child_pid;

child_pid = fork(); // Create a child process

if (child_pid < 0) {
fprintf(stderr, "Fork failed.\n");
return 1;
}
else if (child_pid == 0) {
// Child process
printf("Child process: PID = %d\n", getpid());
printf("Parent process ID = %d\n", getppid());
}
else {
// Parent process
printf("Parent process: PID = %d\n", getpid());
printf("Child process ID = %d\n", child_pid);
}

return 0;
}
```

OUPUT :

```
mint@mint:~/Documents/21BCE1598$ g++ ex5.c
mint@mint:~/Documents/21BCE1598$ ./a.out
Parent process: PID = 3209
Child process ID = 3210
Child process: PID = 3210
Parent process ID = 3209
```

2) **Create Orphan Process program**

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
pid_t pid = fork();

if (pid == 0) {
// Child process
printf("Child process: PID = %d\n", getpid());
printf("Parent process ID: %d\n", getppid());
sleep(5);
printf("New Parent process ID: %d\n", getppid());
} else if (pid > 0) {
// Parent process
printf("Parent process: PID = %d\n", getpid());
exit(0); // Terminate the parent process immediately
} else {
// Error occurred during fork
printf("Fork failed\n");
return 1;
}

return 0;
}
```

OUPUT :

```
mint@mint:~/Documents/21BCE1598$ g++ ex5.c
mint@mint:~/Documents/21BCE1598$ ./a.out
Parent process: PID = 3284
Child process: PID = 3285
Parent process ID: 3284
mint@mint:~/Documents/21BCE1598$ New Parent process ID: 1427
```

3) **Write C program using wait system call.**

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
pid_t pid;
int status;

pid = fork(); // Create a child process

if (pid < 0) {
// Fork failed
perror("fork");
exit(1);
} else if (pid == 0) {
// Child process
printf("Child process executing\n");
sleep(2); // Simulate some work being done by the child process
exit(0);
} else {
// Parent process
printf("Parent process waiting for child to complete\n");
wait(&status); // Wait for the child process to finish
printf("Child process completed\n");
}

return 0;
}
```

OUTPUT:

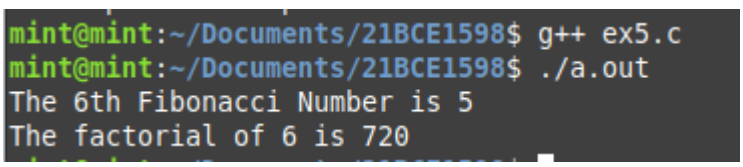
```
mint@mint:~/Documents/21BCE1598$ g++ ex5.c
mint@mint:~/Documents/21BCE1598$ ./a.out
Parent process waiting for child to complete
Child process executing
Child process completed
```

4) **Create a process and compute factorial in child and Fibonacci in parent as executable.**

CODE:

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
void ParentProcess(int n){
int t1 = 0, t2 = 1, next = 0, i;
if(n == 0 || n == 1){
printf("The %dth Fibonacci Number is %d\n", n, n);
}
else{
next = t1 + t2;
}
for (i = 3; i <= n; ++i){
t1 = t2;
t2 = next;
next = t1 + t2;
}
printf("The %dth Fibonacci Number is %d\n", n, t2);
}
void ChildProcess(int n){
int ans = 1;
for (int i=1; i<=n; i++){
ans = ans*i;
}
printf("The factorial of %d is %d\n", n, ans);
}
int main(){
pid_t pid;
pid = fork();
int num = 6;
if (pid==0){
ChildProcess(num);
}
else if (pid>0){
ParentProcess(num);
}
return 1;
}
```

OUPUT:



5) **Create a process and let child do some tasks like computing sum of N numbers.**

CODE:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

int main() {
pid_t pid;
int n, i, sum = 0;

printf("Enter the value of N: ");
scanf("%d", &n);

pid = fork();
if (pid == 0) {
// Child process
for (i = 1; i <= n; i++) {
sum += i;
}
printf("Sum of first %d numbers: %d\n", n, sum);
} else if (pid > 0) {
// Parent process
printf("Parent process is waiting for the child to complete...\n");
wait(NULL);
printf("Child process completed.\n");
} else {
// Fork failed
printf("Fork failed\n");
return 1;
}

return 0;
}
```

OUPUT:

```
mint@mint:~/Documents/21BCE1598$ g++ ex5.c
mint@mint:~/Documents/21BCE1598$ ./a.out
Enter the value of N: 5
Parent process is waiting for the child to complete...
Sum of first 5 numbers: 15
Child process completed.
```

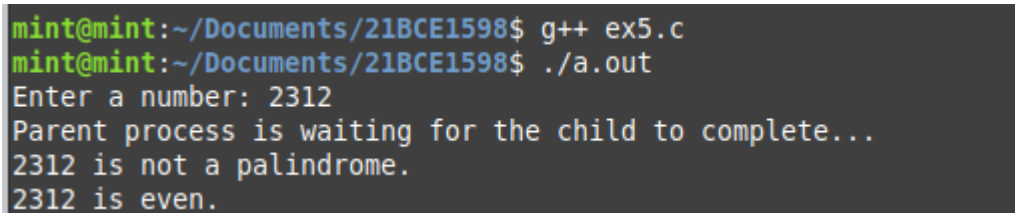
6) **Palindrome and ODD or EVEN as parent and child with Fork.**

CODE:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

int isPalindrome(int num) {
int reversedNum = 0, remainder, originalNum;

originalNum = num;

// Reversing the number
while (num != 0) {
remainder = num % 10;
reversedNum = reversedNum * 10 + remainder;
num /= 10;
}

// Checking if the number is a palindrome
if (originalNum == reversedNum)
return 1;
else
return 0;
}

int main() {
pid_t pid;
int num;

printf("Enter a number: ");
scanf("%d", &num);

pid = fork();
if (pid == 0) {
// Child process
int isPal = isPalindrome(num);
if (isPal)
printf("%d is a palindrome.\n", num);
else
printf("%d is not a palindrome.\n", num);
} else if (pid > 0) {
// Parent process
printf("Parent process is waiting for the child to complete...\n");
wait(NULL);

if (num % 2 == 0)
```

```
printf("%d is even.\n", num);
else
printf("%d is odd.\n", num);
} else {
// Fork failed
printf("Fork failed.\n");
return 1;
}

return 0;
}
```

OUTPUT:



NAME: Krish Jain
REG NO: 21BCE1598

EXP 6

1. FCFS Scheduling

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef struct
{
   int pid;
   int burst_time;
   int waiting_time;
   int turnaround_time;
} Process;

void print_table(Process p[], int n);
void print_gantt_chart(Process p[], int n);

int main()
{
   Process p[MAX];
   int i, j, n;
   int sum_waiting_time = 0, sum_turnaround_time;
   printf("Enter total number of process: ");
   scanf("%d", &n);
   printf("Enter burst time for each process:\n");
   for(i=0; i<n; i++) {
      p[i].pid = i+1;
      printf("P[%d] : ", i+1);
      scanf("%d", &p[i].burst_time);
      p[i].waiting_time = p[i].turnaround_time = 0;
   }

   // calculate waiting time and turnaround time
   p[0].turnaround_time = p[0].burst_time;

   for(i=1; i<n; i++) {
      p[i].waiting_time = p[i-1].waiting_time + p[i-1].burst_time;
      p[i].turnaround_time = p[i].waiting_time + p[i].burst_time;
   }

   // calculate sum of waiting time and sum of turnaround time
   for(i=0; i<n; i++) {
         sum_waiting_time += p[i].waiting_time;
         sum_turnaround_time += p[i].turnaround_time;
```

```
    }

    // print table
    puts(""); // Empty line
    print_table(p, n);
    puts(""); // Empty Line
    printf("Total Waiting Time      : %-2d\n", sum_waiting_time);
    printf("Average Waiting Time    : %-2.2lf\n", (double)sum_waiting_time / (double) n);
    printf("Total Turnaround Time   : %-2d\n", sum_turnaround_time);
    printf("Average Turnaround Time : %-2.2lf\n", (double)sum_turnaround_time / (double)
n);

    // print Gantt chart
    puts(""); // Empty line
    puts("          GANTT CHART          ");
    puts("          ***********          ");
    print_gantt_chart(p, n);
    return 0;
}


void print_table(Process p[], int n)
{
    int i;

    puts("+-----+------------+--------------+-----------------+");
    puts("| PID | Burst Time | Waiting Time | Turnaround Time |");
    puts("+-----+------------+--------------+-----------------+");

    for(i=0; i<n; i++) {
        printf("| %2d  |     %2d     |      %2d      |        %2d       |\n"
              , p[i].pid, p[i].burst_time, p[i].waiting_time, p[i].turnaround_time );
        puts("+-----+------------+--------------+-----------------+");
    }

}


void print_gantt_chart(Process p[], int n)
{
    int i, j;
    // print top bar
    printf(" ");
    for(i=0; i<n; i++) {
        for(j=0; j<p[i].burst_time; j++) printf("--");
        printf(" ");
    }
    printf("\n|");

    // printing process id in the middle
```

```
    for(i=0; i<n; i++) {
        for(j=0; j<p[i].burst_time - 1; j++) printf(" ");
        printf("P%d", p[i].pid);
        for(j=0; j<p[i].burst_time - 1; j++) printf(" ");
        printf("|");
    }
    printf("\n ");
    // printing bottom bar
    for(i=0; i<n; i++) {
        for(j=0; j<p[i].burst_time; j++) printf("--");
        printf(" ");
    }
    printf("\n");

    // printing the time line
    printf("0");
    for(i=0; i<n; i++) {
        for(j=0; j<p[i].burst_time; j++) printf("  ");
        if(p[i].turnaround_time > 9) printf("\b"); // backspace : remove 1 space
        printf("%d", p[i].turnaround_time);

    }
    printf("\n");

}
```

OUTPUT :

```
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BCE1598$ gcc fcsf2.c
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BCE1598$ ./a.out
Enter total number of process: 5
Enter burst time for each process:
P[1] : 12
P[2] : 10
P[3] : 14
P[4] : 22
P[5] : 8

+-----+------------+-------------+-----------------+
| PID | Burst Time | Waiting Time | Turnaround Time |
+-----+------------+-------------+-----------------+
|  1  |     12     |       0     |        12       |
+-----+------------+-------------+-----------------+
|  2  |     10     |      12     |        22       |
+-----+------------+-------------+-----------------+
|  3  |     14     |      22     |        36       |
+-----+------------+-------------+-----------------+
|  4  |     22     |      36     |        58       |
+-----+------------+-------------+-----------------+
|  5  |      8     |      58     |        66       |
+-----+------------+-------------+-----------------+

Total Waiting Time      : 128
Average Waiting Time    : 25.60
Total Turnaround Time   : 194
Average Turnaround Time : 38.80

          GANTT CHART
          ***********
 ---------------------- ------------------- ---------------------------- ---------------------------------------- ----------------
|          P1          |        P2         |            P3              |                P4                        |       P5       |
 ---------------------- ------------------- ---------------------------- ---------------------------------------- ----------------
0                      12                  22                           36                                       58               66
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BCE1598$
```

2. SJF Scheduling

CODE:

```
#include<stdio.h>

int main()
{
   int n,
      process[10],cpu[10],w[10],t[10],At[10],sum_w=0,sum_t=0,i,j,temp=0,temp1=0;
   float avg_w, avg_t;
   printf("enter the number of process\n");
   scanf("%d", &n);
   for(i=0; i<n; i++)
   {
      printf("Enter cpu time of P%d:",i+1);
      scanf("%d", &cpu[i]);
      printf("\n");
   }
   process[0]=1;
   for(i=1; i<n; i++)
   {
      process[i]=i+1;
   }
   for(i=0; i<n; i++)
   {
      for(j=i+1; j<n; j++)
      {
         if(cpu[i]>cpu[j])
         {
            temp=cpu[i];
            cpu[i]=cpu[j];
            cpu[j]=temp;
            temp1=process[i];
            process[i]=process[j];
            process[j]=temp1;
         }
      }
   }
   w[0]=0;
   for(i=1; i<n; i++)
   {
      w[i]=w[i-1]+cpu[i-1];
   }
   for(i=0; i<n; i++)
   {
```

```
        sum_w=sum_w+w[i];
    }

for(i=0; i<n; i++)
    {
        t[i]=w[i]+cpu[i];
        sum_t=sum_t+t[i];
    }
    printf("Process--CPU_time--Wait--Turnaround\n");
for(i=0; i<n; i++)
    {
        printf(" P%d   \t%d   \t%d  \t%d",process[i],cpu[i],w[i],t[i]);
        printf("\n");
    }
    avg_w=(float)sum_w/n;
    avg_t=(float)sum_t/n;
    printf("average waiting time=%.2f\n",avg_w);
    printf("average turnaround time=%.2f\n",avg_t);
    printf("\n");
    printf("==============================GrandChart==================
================\n");
    printf("|");
    for(i=0; i<n; i++)
    {
        printf(" P%d |",process[i]);
    }
    printf("\n0");
    for(i=0; i<n; i++)
    {
        printf("   %d",t[i]);
    }}
```

OUTPUT :

```
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BCE1598$ ./a.out
enter the number of process
5
Enter cpu time of P1:12

Enter cpu time of P2:10

Enter cpu time of P3:14

Enter cpu time of P4:22

Enter cpu time of P5:8

Process--CPU_time--Wait--Turnaround
 P5     8       0       8
 P2     10      8       18
 P1     12      18      30
 P3     14      30      44
 P4     22      44      66
average waiting time=20.00
average turnaround time=33.20

==============================GrandChart==================================
| P5 | P2 | P1 | P3 | P4 |
0   8   18   30   44   66ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BCE1598$
```

3. Priority Scheduling

CODE:

```
#include <stdio.h>

#define MAX_PROCESSES 10

typedef struct {
   int process_id;
   int burst_time;
   int priority;
} Process;

void priorityScheduling(Process processes[], int n) {
   int total_time = 0;
   int waiting_time[MAX_PROCESSES] = {0};
   int turnaround_time[MAX_PROCESSES] = {0};

   // Calculate waiting time and turnaround time for each process
   for (int i = 0; i < n; i++) {
      waiting_time[i] = total_time;
      total_time += processes[i].burst_time;
      turnaround_time[i] = total_time;
   }

   // Calculate average waiting time and turnaround time
   double avg_waiting_time = 0;
   double avg_turnaround_time = 0;
   for (int i = 0; i < n; i++) {
      avg_waiting_time += waiting_time[i];
      avg_turnaround_time += turnaround_time[i];
   }
   avg_waiting_time /= n;
   avg_turnaround_time /= n;

   // Display the table
   printf("Process\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");
   for (int i = 0; i < n; i++) {
      printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].process_id, processes[i].burst_time,
           processes[i].priority, waiting_time[i], turnaround_time[i]);
   }
   printf("Average Waiting Time: %.2f\n", avg_waiting_time);
   printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);

   // Display the Gantt chart
   printf("\nGantt Chart:\n");
   for (int i = 0; i < n; i++) {
      printf("| P%d ", processes[i].process_id);
   }
```

```
    printf("|\n");
    printf("0");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < processes[i].burst_time; j++) {
            printf(" ");
        }
        printf("%2d", turnaround_time[i]);
    }
    printf("\n");
}

int main() {
    int n;
    Process processes[MAX_PROCESSES];

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    printf("Enter burst time and priority for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i].process_id = i + 1;
        printf("Process %d:\n", processes[i].process_id);
        printf("Burst Time: ");
        scanf("%d", &processes[i].burst_time);
        printf("Priority: ");
        scanf("%d", &processes[i].priority);
    }

    priorityScheduling(processes, n);

    return 0;
}e <stdio.h>

#define MAX_PROCESSES 10

typedef struct {
    int process_id;
    int burst_time;
    int priority;
} Process;

void priorityScheduling(Process processes[], int n) {
    int total_time = 0;
    int waiting_time[MAX_PROCESSES] = {0};
    int turnaround_time[MAX_PROCESSES] = {0};

    // Calculate waiting time and turnaround time for each process
    for (int i = 0; i < n; i++) {
        waiting_time[i] = total_time;
        total_time += processes[i].burst_time;
        turnaround_time[i] = total_time;
```

```
    }

    // Calculate average waiting time and turnaround time
    double avg_waiting_time = 0;
    double avg_turnaround_time = 0;
    for (int i = 0; i < n; i++) {
        avg_waiting_time += waiting_time[i];
        avg_turnaround_time += turnaround_time[i];
    }
    avg_waiting_time /= n;
    avg_turnaround_time /= n;

    // Display the table
    printf("Process\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].process_id, processes[i].burst_time,
               processes[i].priority, waiting_time[i], turnaround_time[i]);
    }
    printf("Average Waiting Time: %.2f\n", avg_waiting_time);
    printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);

    // Display the Gantt chart
    printf("\nGantt Chart:\n");
    for (int i = 0; i < n; i++) {
        printf("| P%d ", processes[i].process_id);
    }
    printf("|\n");
    printf("0");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < processes[i].burst_time; j++) {
            printf(" ");
        }
        printf("%2d", turnaround_time[i]);
    }
    printf("\n");
}

int main() {
    int n;
    Process processes[MAX_PROCESSES];

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    printf("Enter burst time and priority for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i].process_id = i + 1;
        printf("Process %d:\n", processes[i].process_id);
        printf("Burst Time: ");
        scanf("%d", &processes[i].burst_time);
        printf("Priority: ");
```

```
        scanf("%d", &processes[i].priority);
    }

    priorityScheduling(processes, n);

    return 0;
}
```

OUTPUT:

```
mint@mint:~/Documents/21BCE1598$ g++ ex6.c
mint@mint:~/Documents/21BCE1598$ ,/a.out
bash: ,/a.out: No such file or directory
mint@mint:~/Documents/21BCE1598$ ./a.out
Enter the number of processes: 5
Enter burst time and priority for each process:
Process 1:
Burst Time: 12
Priority: 3
Process 2:
Burst Time: 14
Priority: 2
Process 3:
Burst Time: 22
Priority: 5
Process 4:
Burst Time: 10
Priority: 1
Process 5:
Burst Time: 8
Priority: 4
Process Burst Time      Priority        Waiting Time    Turnaround Time
1       12              3               0               12
2       14              2               12              26
3       22              5               26              48
4       10              1               48              58
5       8               4               58              66
Average Waiting Time: 28.80
Average Turnaround Time: 42.00

Gantt Chart:
| P1 | P2 | P3 | P4 | P5 |
0           12              26                      48          58        66
mint@mint:~/Documents/21BCE1598$
```

4. Round Robin Scheduling

CODE:

```
#include<stdio.h>

struct times
{
    int p,art,but,wtt,tat,rnt;
};
void sortart(struct times a[],int pro)
{
    int i,j;
    struct times temp;
    for(i=0;i<pro;i++)
    {
        for(j=i+1;j<pro;j++)
        {
            if(a[i].art > a[j].art)
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }}}
    return;
}
int main()
{
    int i,j,pro,time,remain,flag=0,ts;
    struct times a[100];
    float avgwt=0,avgtt=0;
    printf("Round Robin Scheduling Algorithm\n");
    printf("Note -\n1. Arrival Time of at least on process should be 0\n2. CPU should never be idle\n");
    printf("Enter Number Of Processes : ");
    scanf("%d",&pro);
    remain=pro;
    for(i=0;i<pro;i++)
    {
        printf("Enter arrival time and Burst time for Process P%d : ",i);
        scanf("%d%d",&a[i].art,&a[i].but);
        a[i].p = i;
        a[i].rnt = a[i].but;
    }
    sortart(a,pro);
    printf("Enter Time Slice OR Quantum Number : ");
    scanf("%d",&ts);
    printf("\n****************************************\n");
    printf("Gantt Chart\n");
    printf("0");
    for(time=0,i=0;remain!=0;)
    {
        if(a[i].rnt<=ts && a[i].rnt>0)
        {
            time = time + a[i].rnt;
            printf(" -> [P%d] <- %d",a[i].p,time);
            a[i].rnt=0;
            flag=1;
        }
        else if(a[i].rnt > 0)
```

```
        {
            a[i].rnt = a[i].rnt - ts;
            time = time + ts;
            printf(" -> [P%d] <- %d",a[i].p,time);
        }
        if(a[i].rnt==0 && flag==1)
        {
            remain--;
            a[i].tat = time-a[i].art;
            a[i].wtt = time-a[i].art-a[i].but;
            avgwt = avgwt + time-a[i].art-a[i].but;
            avgtt = avgtt + time-a[i].art;
            flag=0;
        }
        if(i==pro-1)
            i=0;
        else if(a[i+1].art <= time)
            i++;
        else
            i=0;
    }
    printf("\n\n");
    printf("****************************************\n");
    printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\n");
    printf("****************************************\n");
    for(i=0;i<pro;i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\n",a[i].p,a[i].art,a[i].but,a[i].tat,a[i].wtt);
    }
    printf("****************************************\n");
    avgwt = avgwt/pro;
    avgtt = avgtt/pro;
    printf("Average Waiting Time : %.2f\n",avgwt);
    printf("Average Turnaround Time : %.2f\n",avgtt);
    return 0;
}
```

OUTPUT:

```
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop/21BCE1598$ ./a.out
Round Robin Scheduling Algorithm
Note -
1. Arrival Time of at least on process should be 0
2. CPU should never be idle
Enter Number Of Processes : 5
Enter arrival time and Burst time for Process P0 : 0 12
Enter arrival time and Burst time for Process P1 : 0 14
Enter arrival time and Burst time for Process P2 : 0 22
Enter arrival time and Burst time for Process P3 : 0 10
Enter arrival time and Burst time for Process P4 : 0 8
Enter Time Slice OR Quantum Number : 4

**************************************
Gantt Chart
0 -> [P0] <- 4 -> [P1] <- 8 -> [P2] <- 12 -> [P3] <- 16 -> [P4] <- 20 -> [P0] <- 24 -> [P1] <- 28 -> [P2] <- 32 -> [P3] <- 36 -> [P4] <- 40 -> [P0] <- 44 -> [P1] <- 48 -
> [P2] <- 52 -> [P3] <- 54 -> [P1] <- 56 -> [P2] <- 60 -> [P2] <- 64 -> [P2] <- 66

**************************************
Pro     ArTi    BuTi    TaTi    WtTi
**************************************
P0      0       12      44      32
P1      0       14      56      42
P2      0       22      66      44
P3      0       10      54      44
P4      0       8       40      32
**************************************
Average Waiting Time : 38.80
Average Turnaround Time : 52.00
```

4. Primitive SJF Scheduling

CODE:

```
#include<stdio.h>

int main(){
    int i,j,k,p,s=0, get=0, idle=0, t_burst, t_row, pre_process_row, final=0;
    float sum=0;

    printf("Please enter the number process : ");
    scanf("%d",&p);

    int a[p][5];
    int b[p][5];

    printf("\nProcess\tArrival\tBurst\n-------\t-------\t-----\n");
    for(i=0;i<p;i++){
        for(j=0;j<3;j++){
            scanf("%d",&a[i][j]);
            }
        a[i][3]=a[i][2];
        }

    printf("\n\nTime-Line is as follows (Verticle View)....\n\n");

    i=a[0][1];
    while(final!=p){
        get=0;
        k=0;
        while(k<p){
            if(a[k][1]<=i){
                if(a[k][2]!=0){
                    get=1;
                    t_burst=a[k][2];
                    t_row=k;
                    idle=0;
                    break;
                    }
                else
                    k++;
                }
            else{
                if(idle==0)
                    printf("%5d----------\n        |Idle  |\n",i);
                idle=1;
                break;
                }
            }
        if(get!=0){
            k=0;
            while(a[k][1]<=i && k<p){
                if(a[k][2]!=0){
                    if(t_burst>a[k][2]){
                        t_burst=a[k][2];
                        t_row=k;
                        }
                    }
```

```
            k++;
            }

        a[t_row][2]-=1;

        if(i==a[0][1])
            printf("%5d----------\n        |p-%-4d|\n",i,a[t_row][0]);
        else{
            if(pre_process_row!=t_row)
                printf("%5d----------\n        |p-%-4d|\n",i,a[t_row][0]);
            }

        pre_process_row=t_row;

        if(a[t_row][2]==0){
            final++;
            b[s][0]=a[t_row][0];
            b[s][1]=a[t_row][1];
            b[s][2]=i;
            b[s][3]=a[t_row][3];
            b[s][4]=((i-a[t_row][1])-a[t_row][3])+1;
            sum+=((i-a[t_row][1])-a[t_row][3])+1;
            s++;
            }
        }
    i++;
    }
printf("%5d----------\n",i)

printf("Table of processes with completion record as they were completed\n\n");
printf("\n\nProcess\tArrival\tFin\tTotal\tWait\n-------\t-------\t---\t-----\t----\n");

for(i=0;i<s;i++)
    printf("%d\t%d\t%d\t%d\t%d\n", b[i][0], b[i][1], b[i][2], b[i][3], b[i][4]);
printf("\nAvg. Wait time = %f",sum/p);
printf("\n\n");

return 0;
}
```

OUTPUT:

```
ex2@ilab-HP-Desktop-Pro-G2:~$ gcc psjf.c
ex2@ilab-HP-Desktop-Pro-G2:~$ ./a.out
Please enter the number process : 5

Process Arrival Burst
------- ------- -----
1 0 12
2 0 10
3 0 14
4 0 22
5 0 8


Time-Line is as follows (Verticle View)....

    0-----------
        |p-5   |
    8-----------
        |p-2   |
   18-----------
        |p-1   |
   30-----------
        |p-3   |
   44-----------
        |p-4   |
   66-----------
Table of processes with completion record as they were completed



Process Arrival Fin     Total   Wait
------- ------- ---     -----   ----
5       0       7       8       0
2       0       17      10      8
1       0       29      12      18
3       0       43      14      30
4       0       65      22      44

Avg. Wait time = 20.000000
```

4. Primitive Priority Scheduling

CODE:

```
#include<stdio.h>

// Structure to store process information
struct Process {
    int pid;           // Process ID
    int burst_time;    // Burst time of process
    int priority;      // Priority of process
    int arrival_time;  // Arrival time of process
    int waiting_time;  // Waiting time of process
    int turnaround_time;// Turnaround time of process
};

// Function to sort processes by priority using bubble sort
void sort_processes_by_priority(struct Process p[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].priority > p[j+1].priority) {
                struct Process temp = p[j];
                p[j] = p[j+1];
                p[j+1] = temp;
            }}}}

// Function to calculate waiting time and turnaround time for each process
void calculate_waiting_time_and_turnaround_time(struct Process p[], int n) {
    int total_waiting_time = 0;
    int total_turnaround_time = 0;
    int current_time = 0;

    printf("\nProcess\tBurst Time\tPriority\tArrival Time\tWaiting Time\tTurnaround Time\n");

    // Calculate waiting time and turnaround time for each process
    for (int i = 0; i < n; i++) {
        // Update current time to the arrival time of next process
        if (current_time < p[i].arrival_time) {
            current_time = p[i].arrival_time;
        }

        // Calculate waiting time for current process
        p[i].waiting_time = current_time - p[i].arrival_time;

        // Calculate turnaround time for current process
        p[i].turnaround_time = p[i].burst_time + p[i].waiting_time;

        // Update total waiting time and total turnaround time
        total_waiting_time += p[i].waiting_time;
        total_turnaround_time += p[i].turnaround_time;

        // Print process information
        printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burst_time, p[i].priority, p[i].arrival_time,
p[i].waiting_time, p[i].turnaround_time);
```

```
        // Update current time to the completion time of current process
        current_time += p[i].burst_time;
    }

    // Print average waiting time and average turnaround time
    float avg_waiting_time = (float) total_waiting_time / n;
    float avg_turnaround_time = (float) total_turnaround_time / n;
    printf("\nAverage Waiting Time: %.2f", avg_waiting_time);
    printf("\nAverage Turnaround Time: %.2f", avg_turnaround_time);
}

// Function to print Gantt chart
void printGanttChart(struct Process processes[], int n) {
    int i, j, time = 0;
    printf("\n");
    for (i = 0; i < n; i++) {
        printf(" ");
        for (j = 0; j < processes[i].burst_time; j++) {
            printf("--");
        }
        printf(" ");
    }
    printf("\n|");
    for (i = 0; i < n; i++) {
        for (j = 0; j < processes[i].burst_time - 1; j++) {
            printf(" ");
        }
        printf("P%d", processes[i].pid);
        for (j = 0; j < processes[i].burst_time - 1; j++) {
            printf(" ");
        }
        printf("|");
    }
    printf("\n ");
    for (i = 0; i < n; i++) {
        printf(" ");
        for (j = 0; j < processes[i].burst_time; j++) {
            printf("--");
        }
        printf(" ");
    }
    printf("\n");
    printf("0");
    for (i = 0; i < n; i++) {
        time += processes[i].burst_time;
        printf("      %d", time);
    }
    printf("\n");
}



int main() {
    // Take input from user
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
```

```
    struct Process p[n];
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for process %d:\n", i+1);
        printf("Process ID: ");
        scanf("%d", &p[i].pid);
        printf("Burst Time: ");
        scanf("%d", &p[i].burst_time);
        printf("Priority: ");
        scanf("%d", &p[i].priority);
        printf("Arrival Time: ");
        scanf("%d", &p[i].arrival_time);
    }

    // Sort processes by priority
    sort_processes_by_priority(p, n);

    // Calculate waiting time and turnaround time for each process
    calculate_waiting_time_and_turnaround_time(p, n);

    // Print Gantt chart
    printGanttChart(p, n);

    return 0;
}
```

OUTPUT:

```
ex2@ilab-HP-Desktop-Pro-G2:~$ gcc psjf.c
ex2@ilab-HP-Desktop-Pro-G2:~$ ./a.out
Enter the number of processes: 5

Enter details for process 1:
Process ID: 1
Burst Time: 12
Priority: 3
Arrival Time: 0

Enter details for process 2:
Process ID: 2
Burst Time: 10
Priority: 1
Arrival Time: 2

Enter details for process 3:
Process ID: 3
Burst Time: 22
Priority: 2
Arrival Time: 5

Enter details for process 4:
Process ID: 4
Burst Time: 8
Priority: 5
Arrival Time: 7

Enter details for process 5:
Process ID: 5
Burst Time: 14
Priority: 4
Arrival Time: 9
```

```
Enter details for process 5:
Process ID: 5
Burst Time: 14
Priority: 4
Arrival Time: 9

Process Burst Time      Priority        Arrival Time    Waiting Time    Turnaround Time
P2              10              1               2               0               10
P3              22              2               5               7               29
P1              12              3               0               34              46
P5              14              4               9               37              51
P4              8               5               7               53              61

Average Waiting Time: 26.20
Average Turnaround Time: 39.40
 ------------------  ----------------------------------------  ----------------------  --------------------------  ---------------
|        P2        |                  P3                    |          P1           |            P5             |      P4       |
  ------------------  ----------------------------------------  ----------------------  --------------------------  ---------------
0      10      32      44      58      66
ex2@ilab-HP-Desktop-Pro-G2:~$
```

NAME: Krish Jain
REG NO: 21BCE1598

EXP 7

1. Deadlock Detection

CODE:

```
#include <stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n, m, i, j, k;
void input() {
  printf("Enter the no of Processes: ");
  scanf("%d", & n);
  printf("Enter the no of resource instances: ");
  scanf("%d", & m);
  printf("Enter the Max Matrix\n");
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
      scanf("%d", & max[i][j]);
    }
  }
  printf("Enter the Allocation Matrix\n");
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
```

```
      scanf("%d", & alloc[i][j]);
    }
  }
  printf("Enter the available Resources\n");
  for (j = 0; j < m; j++) {
    scanf("%d", & avail[j]);
  }
}
void show() {
  int i, j;
  printf("Process\t Allocation\t Max\t Available\t");
  for (i = 0; i < n; i++) {
    printf("\nP%d\t ", i + 1);
    for (j = 0; j < m; j++) {
      printf("%d ", alloc[i][j]);
    }
    printf("\t\t");
    for (j = 0; j < m; j++) {
      printf("%d ", max[i][j]);
    }
    printf("\t ");
    if (i == 0) {
      for (j = 0; j < m; j++) {
        printf("%d ", avail[j]);
      }
```

```
    }
  }
}
void printTotalAvailableResources() {
  int totalAvailable[100] = {
    0
  };
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
      totalAvailable[i] += alloc[j][i];
    }
    totalAvailable[i] = avail[i] + totalAvailable[i];
  }
  printf("\n\nTotal Available Resources: ");
  for (i = 0; i < m; i++) {
    printf("%d ", totalAvailable[i]);
  }
  printf("\n");
}
int main() {
  printf("********** Deadlock Detection Algorithm **********\n");
  input();
  show();
  int f[n], ans[n], ind = 0;
  for (k = 0; k < n; k++) {
```

```
  f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++) {
  for (j = 0; j < m; j++) {
    need[i][j] = max[i][j] - alloc[i][j];
  }
}
int y = 0;
for (k = 0; k < 5; k++) {
  for (i = 0; i < n; i++) {
    if (f[i] == 0) {
      int flag = 0;
      for (j = 0; j < m; j++) {
        if (need[i][j] > avail[j]) {
          flag = 1;
          break;
        }
      }
      if (flag == 0) {
        ans[ind++] = i;
        for (y = 0; y < m; y++) {
          avail[y] += alloc[i][y];
        }
        f[i] = 1;
```

```
        }
      }
    }
  }
  int flag = 1;
  for (int i = 0; i < n; i++) {
    if (f[i] == 0) {
      flag = 0;
      printf("\nDeadlock detected!\n");
      printf("The following system is not safe\n");
      break;
    }
  }
  if (flag == 1) {
    printf("\nFollowing is the SAFE Sequence\n");
    for (i = 0; i < n - 1; i++) {
      printf("P%d -> ", ans[i]);
    }
    printf("P%d\n", ans[n - 1]);
  }
  printTotalAvailableResources();
  return 0;
}
```

OUTPUT:

```
ex2 ilab-HP-Desktop-Pro-G2 ~/Documents/21BCE1598 ./a.out
********** Deadlock Detection Algorithm ************
Enter the no of Processes: 3
Enter the no of resource instances: 3
Enter the Max Matrix
3 6 8
4 3 3
3 4 4
Enter the Allocation Matrix
3 3 3
2 0 3
1 2 4
Enter the available Resources
1 2 0
Process  Allocation      Max     Available
P1       3 3 3          3 6 8    1 2 0
P2       2 0 3          4 3 3
P3       1 2 4          3 4 4
Deadlock detected!
The following system is not safe
```

```
ex2 ilab-HP-Desktop-Pro-G2 ~/Documents/21BCE1598 g++ tt.c

ex2 ilab-HP-Desktop-Pro-G2 ~/Documents/21BCE1598 ./a.out
********** Deadlock Detection Algorithm ************
Enter the no of Processes: 3
Enter the no of resource instances: 3
Enter the Max Matrix
1 1 1
2 2 2
3 3 3
Enter the Allocation Matrix
2 2 2
1 1 1
3 3 3
Enter the available Resources
3 3 2
Process  Allocation      Max     Available
P1       2 2 2          1 1 1    3 3 2
P2       1 1 1          2 2 2
P3       3 3 3          3 3 3
Following is the SAFE Sequence
P0 -> P1 -> P2


Total Available Resources: 3 3 2
```

NAME: Krish Jain
REG NO: 21BCE1598

EXP 8

1. Producer Consumer Problem Using Semaphores

CODE:

```
#include<stdio.h>
#include<stdlib.h>
int mutex=1,full=0,empty,x=0;
char a[100];
int main()
{
int n;
printf("Enter Buffer Size: ");
scanf("%d",&empty);
int j=empty;
void producer();
void consumer();
int wait(int);
int signal(int);
printf("\n1.PRODUCER\n2.CONSUMER\n3.DISPLAY\n4.EXIT\n");
while(1) {
printf("\nENTER YOUR CHOICE\n");
scanf("%d",&n);
switch(n)
{ case 1:
if((mutex==1)&&(empty!=0)){
printf("\nWhat to Produce\n");
scanf("%s",a);
producer(a);
}
else
printf("BUFFER IS FULL\n");
break;
case 2:
if((mutex==1)&&(full!=0))
consumer(a);
else
printf("BUFFER IS EMPTY\n");
break;
case 3:
printf("Buffer Size: %d\n",j-empty);
break;
case 4:
exit(0);
break;
```

```
}}}
int wait(int s) {
return(--s); }

int signal(int s) {
return(++s); }

void producer(char a[]) {
mutex=wait(mutex);
full=signal(full);
empty=wait(empty);
x++;
printf("producer produces item%d: %s\n",x,a);
mutex=signal(mutex); }

void consumer(char a[]) {
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("consumer consumes item%d: %s\n",x,a);
x--;
mutex=signal(mutex); }
```

OUTPUT:

```
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop$ gcc hi.c
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop$ ./a.out
Enter Buffer Size: 3

1.PRODUCER
2.CONSUMER
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE
1

What to Produce
Pasta
producer produces item1: Pasta

ENTER YOUR CHOICE
2
consumer consumes item1: Pasta

ENTER YOUR CHOICE
3
Buffer Size: 0
```

```
ENTER YOUR CHOICE
2
BUFFER IS EMPTY

ENTER YOUR CHOICE
1

What to Produce
Pasta
producer produces item1: Pasta

ENTER YOUR CHOICE
1

What to Produce
Eggs
producer produces item2: Eggs

ENTER YOUR CHOICE
1

What to Produce
Chicken
producer produces item3: Chicken
```

```
ENTER YOUR CHOICE
1
BUFFER IS FULL

ENTER YOUR CHOICE
3
Buffer Size: 3

ENTER YOUR CHOICE
4
ex2@ilab-HP-Desktop-Pro-G2:~/Desktop$
```

2. Readers Writers Problem Using Semaphores

CODE:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

sem_t mutex;    // Semaphore for mutual exclusion
sem_t db;       // Semaphore for controlling database access
int readercount = 0;    // Counter for the number of active readers

void *reader(void *arg);
void *writer(void *arg);

int main()
{
   pthread_t reader1, reader2, writer1, writer2;

   sem_init(&mutex, 0, 1);
   sem_init(&db, 0, 1);

   // Create reader and writer threads
   pthread_create(&reader1, NULL, reader, (void *)1);
   pthread_create(&reader2, NULL, reader, (void *)2);
   pthread_create(&writer1, NULL, writer, (void *)1);
   pthread_create(&writer2, NULL, writer, (void *)2);

   // Wait for threads to finish
   pthread_join(reader1, NULL);
   pthread_join(reader2, NULL);
   pthread_join(writer1, NULL);
   pthread_join(writer2, NULL);

   sem_destroy(&mutex);
   sem_destroy(&db);

   return 0;
}

void *reader(void *arg)
```

```
{
    int readerID = (int)arg;

    while (1)
    {
        // Acquire mutex to update reader count
        sem_wait(&mutex);
        readercount++;
        if (readercount == 1)
        {
            // First reader, acquire database
            sem_wait(&db);
        }
        sem_post(&mutex);

        // Reader reading
        printf("Reader %d is reading\n", readerID);
        // Reading is performed here

        // Acquire mutex to update reader count
        sem_wait(&mutex);
        readercount--;
        if (readercount == 0)
        {
            // Last reader, release database
            sem_post(&db);
        }
        sem_post(&mutex);

        // Reader completes reading
        printf("Reader %d completed reading\n", readerID);

        // Sleep for a while before next read
        sleep(1);
    }

    pthread_exit(NULL);
}

void *writer(void *arg)
```

```
{
    int writerID = (int)arg;

    while (1)
    {
        // Writer waiting
        printf("Writer %d is waiting\n", writerID);

        // Acquire database
        sem_wait(&db);

        // Writer writing
        printf("Writer %d is writing\n", writerID);
        // Writing is performed here

        // Release database
        sem_post(&db);

        // Writer completes writing
        printf("Writer %d completed writing\n", writerID);

        // Sleep for a while before next write
        sleep(1);
    }

    pthread_exit(NULL);
}
```

OUTPUT:

```
mint@mint:~/Desktop$ cc 2.c -lpthread
2.c: In function ‘reader’:
2.c:39:20: warning: cast from pointer to integer of different size [-Wpointer-to
-int-cast]
   39 |     int readerID = (int)arg;
      |                    ^
2.c:71:9: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function
-declaration]
   71 |         sleep(1);
      |         ^~~~~
2.c: In function ‘writer’:
2.c:79:20: warning: cast from pointer to integer of different size [-Wpointer-to
-int-cast]
   79 |     int writerID = (int)arg;
      |                    ^
```

```
mint@mint:~/Desktop$ ./a.out
Reader 2 is reading
Reader 2 completed reading
Reader 1 is reading
Reader 1 completed reading
Writer 2 is waiting
Writer 2 is writing
Writer 2 completed writing
Writer 1 is waiting
Writer 1 is writing
Writer 1 completed writing
Reader 2 is reading
Reader 2 completed reading
Reader 1 is reading
Reader 1 completed reading
Writer 2 is waiting
Writer 2 is writing
Writer 2 completed writing
Writer 1 is waiting
Writer 1 is writing
Writer 1 completed writing
Reader 2 is reading
```

NAME: Krish Jain
REG NO: 21BCE1598

EXP 9

Memory Allocation

1. FIRST FIT:

CODE:

```
#include<stdio.h>
#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
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]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1) //if bf[j] is not allocated
{
temp=b[j]-f[i];
if(temp>=0)
if(highest<temp)
{
ff[i]=j;
highest=temp;
}
}
}
frag[i]=highest;
```

```
bf[ff[i]]=1;
highest=0;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

OUTPUT:

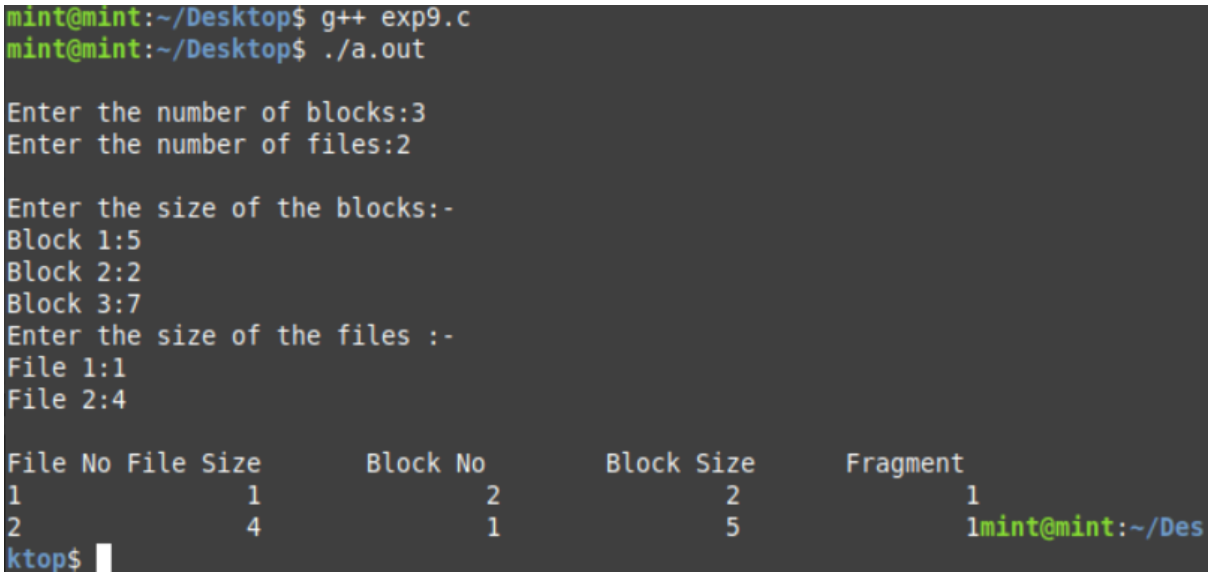
2. BEST FIT:

CODE:

```
#include<stdio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("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]);
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
```

```
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
if(lowest>temp)
{
ff[i]=j;
lowest=temp;
}
}}
frag[i]=lowest; bf[ff[i]]=1; lowest=10000;
}
printf("\nFile No\tFile Size \tBlock No\tBlock
Size\tFragment"); for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

OUTPUT:



3. WORST FIT:

CODE:

```
#include<stdio.h>
#define max 25
int main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - First Fit");
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]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
{
ff[i]=j;
break;
}
}
}
frag[i]=temp;
bf[ff[i]]=1;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
}
```

OUTPUT:

```
mint@mint:~/Desktop$ g++ exp9.c
mint@mint:~/Desktop$ ./a.out

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               1               1               5               4
2               4               3               7               3mint@mint:~/Des
ktop$
```

NAME: Krish Jain
REG NO: 21BCE1598

EXP 10

Page Replacement Algorithm

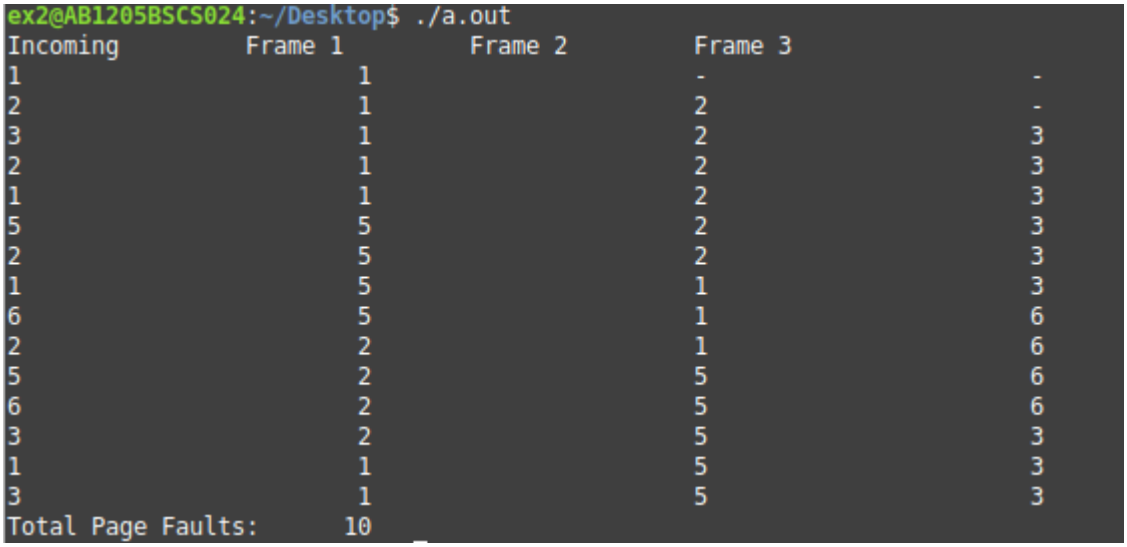
1. FIFO:

CODE:

```
// C program for FIFO page replacement algorithm
#include<stdio.h>
int main()
{
   int incomingStream[] = {1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3};
   int pageFaults = 0;
   int frames = 3;
   int m, n, s, pages;

   pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

   printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");
   int temp[frames];
   for(m = 0; m < frames; m++)
   {
      temp[m] = -1;
   }

   for(m = 0; m < pages; m++)
   {
      s = 0;

      for(n = 0; n < frames; n++)
      {
         if(incomingStream[m] == temp[n])
         {
            s++;
            pageFaults--;
         }
      }
      pageFaults++;

      if((pageFaults <= frames) && (s == 0))
      {
         temp[m] = incomingStream[m];
      }
      else if(s == 0)
      {
```

```
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }

        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
    }

    printf("\nTotal Page Faults:\t%d\n", pageFaults);
    return 0;
}
```

OUTPUT:



2. LRU:

CODE:

```
// C program for LRU page replacement algorithm
#include<stdio.h>
#include<limits.h>

int checkHit(int incomingPage, int queue[], int occupied){

    for(int i = 0; i < occupied; i++){
        if(incomingPage == queue[i])
            return 1;
    }

    return 0;
}

void printFrame(int queue[], int occupied)
{
    for(int i = 0; i < occupied; i++)
        printf("%d\t\t\t",queue[i]);
}

int main()
{

//    int incomingStream[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1};
//    int incomingStream[] = {1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3};
    int incomingStream[] = {1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3};

    int n = sizeof(incomingStream)/sizeof(incomingStream[0]);
    int frames = 3;
    int queue[n];
    int distance[n];
    int occupied = 0;
    int pagefault = 0;

    printf("Page\t Frame1 \t Frame2 \t Frame3\n");

    for(int i = 0;i < n; i++)
    {
        printf("%d:  \t\t",incomingStream[i]);
        // what if currently in frame 7
        // next item that appears also 7
        // didnt write condition for HIT

        if(checkHit(incomingStream[i], queue, occupied)){
            printFrame(queue, occupied);
```

```
        }

        // filling when frame(s) is/are empty
        else if(occupied < frames){
            queue[occupied] = incomingStream[i];
            pagefault++;
            occupied++;

            printFrame(queue, occupied);
        }
        else{

            int max = INT_MIN;
            int index;
            // get LRU distance for each item in frame
            for (int j = 0; j < frames; j++)
            {
                distance[j] = 0;
                // traverse in reverse direction to find
                // at what distance  frame item occurred last
                for(int k = i - 1; k >= 0; k--)
                {
                    ++distance[j];

                    if(queue[j] == incomingStream[k])
                        break;
                }

                // find frame item with max distance for LRU
                // also notes the index of frame item in queue
                // which appears furthest(max distance)
                if(distance[j] > max){
                    max = distance[j];
                    index = j;
                }
            }
            queue[index] = incomingStream[i];
            printFrame(queue, occupied);
            pagefault++;
        }

        printf("\n");
    }

    printf("Page Fault: %d",pagefault);

    return 0;
}
```

OUTPUT:

```
ex2@AB1205BSCS024:~/Desktop$ ./a.out
Page     Frame1          Frame2          Frame3
1:               1
2:               1                       2
3:               1                       2                       3
2:               1                       2                       3
1:               1                       2                       3
5:               1                       2                       5
2:               1                       2                       5
1:               1                       2                       5
6:               1                       2                       6
2:               1                       2                       6
5:               5                       2                       6
6:               5                       2                       6
3:               5                       3                       6
1:               1                       3                       6
3:               1                       3                       6
Page Fault: 8ex2@AB1205BSCS024:~/Desktop$
```

3. OPTIMAL:

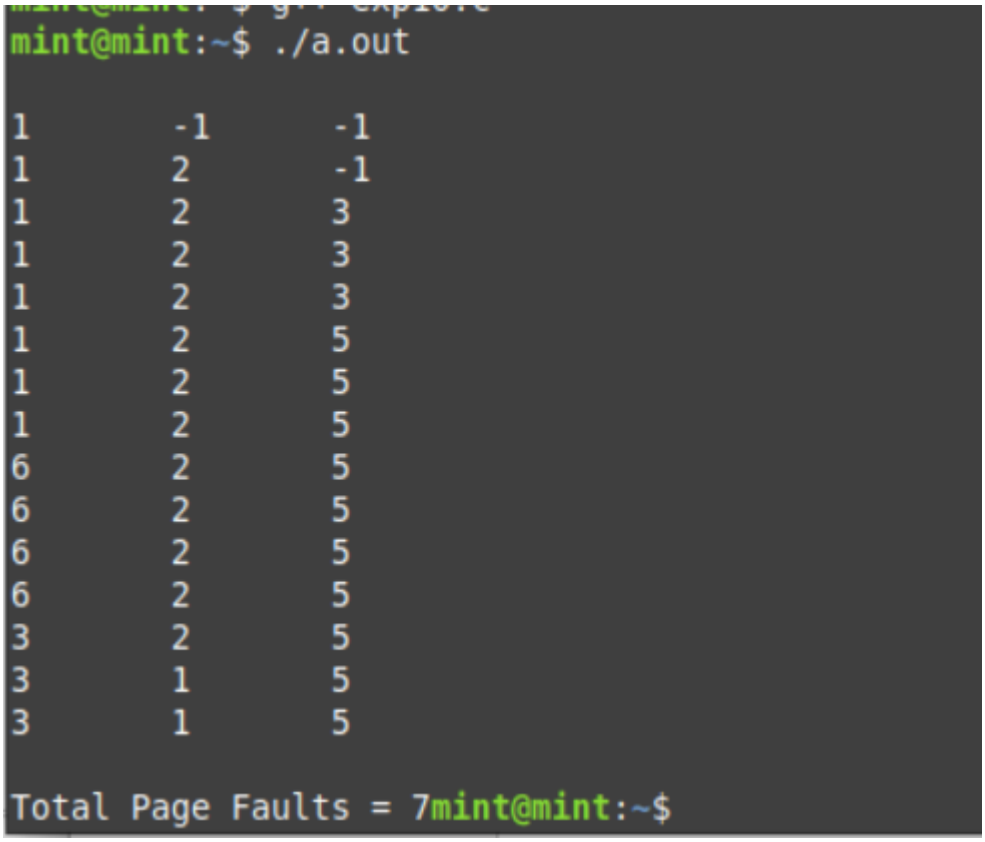
CODE:

```
#include<stdio.h>
int main()
{
   int no_of_frames=3, no_of_pages=15, frames[10], temp[10], flag1, flag2, flag3, i, j, k, pos,
max, faults = 0;


   int pages[30]={1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3};



   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:



RESULT: Therefore, Optimal Page Replacement Algorithm is the Best as it has the least Page Fault.