Department of Computer Applications

Programme: MCA

Course Code : CA704

Course Name OPERATING SYSTEMS LAB

MCA BATCH 2020-23

SEMESTER – 2

Source Code & Output of Cycle Sheet-I & Cycle Sheet-II

Name Ramratan Sharma

Roll number 205120081

```
205120081@ca: ~
205120081@ca:~$ ls
a.out           ques15.sh   ques26.c                  ques33_dining.c
data            ques16.sh   ques27.c                  ques34_best.c
file.txt        ques17.c    ques28_fcfs.c             ques34_first.c
listforques21   ques17.sh   ques28_priority.c         ques34_worst.c
newpro          ques18.c    ques28_rr.c               ques35.c
nitt.txt        ques19.c    ques28_sjf.c              ques3.sh
q16.sh          ques20.c    ques29.c                  ques4.sh
q17.c           ques21.c    ques2.sh                  ques5.sh
queq17.c        ques22_b.c  ques30_binarySemaphore.c  ques6.sh
ques0.sh        ques22.c    ques30_counting.c         ques7.sh
ques10.sh       ques23.c    ques31.c                  ques8.sh
ques11.sh       ques24.c    ques31_consumer.c         ques9.sh
ques12.sh       ques25.c    ques32_reader.c           tmp
ques13.sh       ques25_r.c  ques32_writer.c
ques14.sh       ques25_w.c  ques33_bankers.c
205120081@ca:~$
```

CYCLE SHEET – I

Question 2 : Write a shell script to read three numbers from standard input and print the minimum value and maximum.

Answer

```
echo "Enter three nummbers as input  "
read a b c
if [ $a -gt $b ] && [ $a -gt $c ]
then
   echo "$a is the greatest"
   if [ $b -gt $c ]
   then
      echo "$c is smallest"
   else
      echo "$b is smallest"
   fi
elif [ $b -gt $a ] && [ $b -gt $c ]
then
   echo "$b is greatest"
   if [ $a -gt $c ]
   then
      echo "$c is smallest"
   else
      echo "$a is smallest"
   fi
else
   echo "$c is greatest"
   if [ $a -gt $b ]
   then
      echo "$b is smallest"
   else
      echo "$a is smallest"
   fi
fi
```

Output

```
205120081@ca: ~
205120081@ca:~$ sh ques2.sh
Enter three nummbers as input
45 30 65
65 is greatest
30 is smallest
205120081@ca:~$
```

Question 3 : Write a shell script to swap two numbers without using 3rd variable

Answer

```
echo "Enter two numbers to swap"
read a b
echo "Before swapping a=$a, b=$b"
a=$((a^b))
b=$((a^b))
a=$((a^b))

echo "After swapping a=$a, b=$b"
```

Output

```
205120081@ca: ~
205120081@ca:~$ sh ques3.sh
Enter two numbers to swap
45 30
Before swapping a=45, b=30
After swapping a=30, b=45
205120081@ca:~$
```

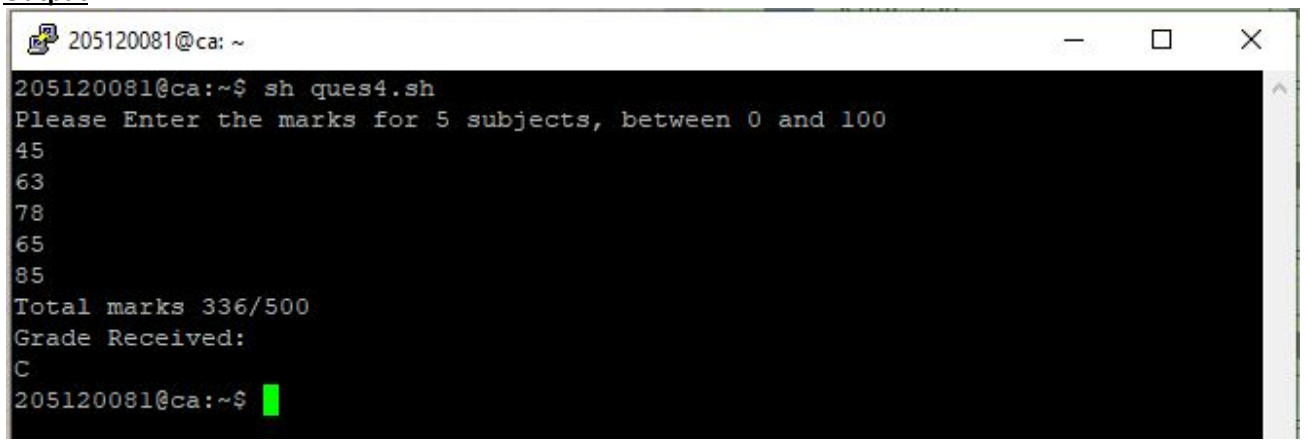
Question 4 : Write a shell script to read the marks of a Student and print the grade.

Answer

```
echo "Please Enter the marks for 5 subjects, between 0 and 100"
read a
read b
read c
read d
read e
sum=$((a+b+c+d+e))

echo "Total marks $sum/500 "
echo "Grade Received: "
if [ $sum -gt 200 ]
then
   if [ $sum -gt 250 ]
   then
      if [ $sum -gt 300 ]
```

```
      then
        if [ $sum -gt 350 ]
        then
          if [ $sum -gt 400 ]
          then
            if [ $sum -gt 450 ]
            then
              echo "A"
            else
              echo "B1"
            fi
          else
            echo "B2"
          fi
        else
          echo "C"
        fi
      else
        echo "D"
      fi
    else
      echo "E"
    fi
else
    echo "F"
fi
```

Output



Question 5 : Write a shell script to read two integer numbers and perform basic arithmetic operations based on user's choice (use 'case' structure).

Answer

```
# !/bin/bash

# Take user Input
echo "Enter Two numbers : "
read a
read b

# Input type of operation
echo "Enter Choice :"
echo "1. Addition"
```

```
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
echo "Enter Your Choice"
read ch

# Switch Case to perform
# calulator operations
case $ch in
  1)res=`echo $a + $b | bc`
  ;;
  2)res=`echo $a - $b | bc`
  ;;
  3)res=`echo $a \* $b | bc`
  ;;
  4)res=`echo "scale=2; $a / $b" | bc`
  ;;
esac
echo "Result : $res"
```

Output

```
205120081@ca: ~
205120081@ca:~$ ls
10.sh  6.sh  9.sh      ques2.sh  ques4.sh
11.sh  8.sh  ques0.sh  ques3.sh  ques5.sh
205120081@ca:~$ sh ques5.sh
Enter Two numbers :
10
15
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter Your Choice
1
Result : 25
205120081@ca:~$ sh ques5.sh
Enter Two numbers :
13
14
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter Your Choice
3
Result : 182
205120081@ca:~$
```

Question 6 : Write a shell script to find the sum of first ‘N’ Natural Numbers (use ‘while’ structure)

Answer

```
echo "Enter N :"
read n
i=0
while [ $i -lt $n ]
do
        echo -n "Enter Number :"
        read val
        sum=`expr $sum + $val`
```

```
        i=`expr $i + 1`
done
echo "Sum of Enter Numbers = $sum"
```

Output

```
205120081@ca: ~
205120081@ca:~$ sh ques6.sh
Enter N :
5
Enter Number :1
Enter Number :2
Enter Number :3
Enter Number :4
Enter Number :5
Sum of Enter Numbers = 15
205120081@ca:~$
```

Question 7 : Write a shell script to find the sum of first 'N' numbers in Fibonacci series (use 'for' structure)

Answer

```
echo "Enter Value of n:"
read N
a=0


b=1
sum=0
for (( i=0; i<N; i++ ))
do
  echo -n "$sum "
  fn=$((a + b))
  a=$b
  b=$fn
  sum=$((sum+a))
done
echo "Sum = $sum"
```

Output

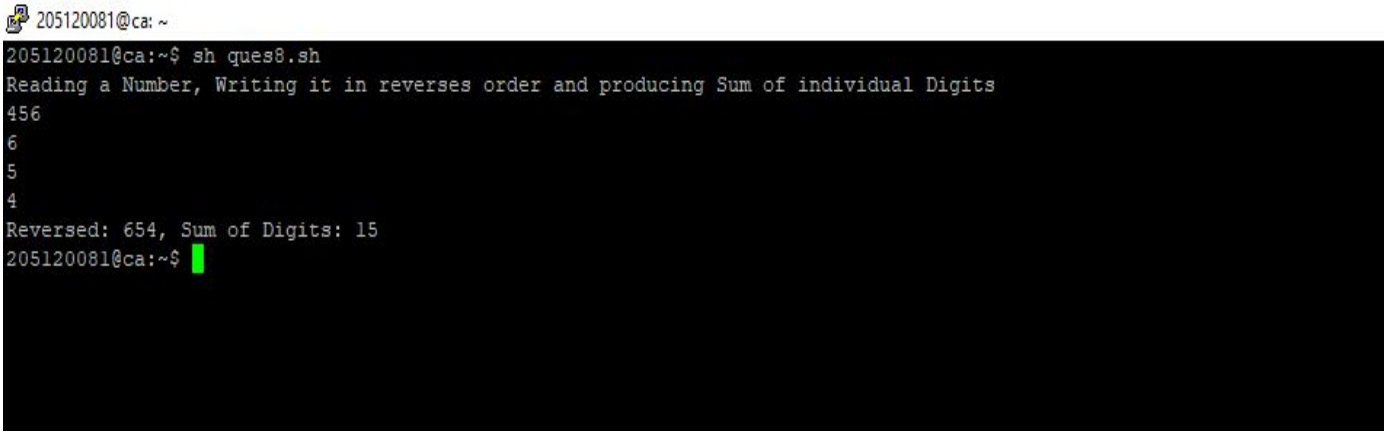
```
205120081@ca:~$ sh 7.sh
Enter value of n:
10
0
1
1
2
3
5
8
13
21
34
Sum = 88
205120081@ca:~$
```

Question 8 : Write a shell script to print a given number in reverse order and sum of the individual digits.

Answer

```
echo "Reading a Number, Writing it in reverses order and producing Sum of individual Digits"
read num
rem=0
sm=0
revdig=0
while [ $num -gt 0 ]
do
   rem=`expr $num % 10`
   echo $rem
   sm=$((sm+rem))
   num=$((num/10))
   revdig=$((revdig*10+rem))

done
echo "Reversed: $revdig, Sum of Digits: $sm"
```

Output



Question 9 : Write a shell script to read two strings and display whether it is equal, not equal, null strings or string with special characters.

Answer

```
#!bin/sh
read -p " enter string 1 : " str1
read -p " enter string 2 : " str2
if [ $str1 = $str2 ]
then echo " strings are equal !"
else
        echo " strings are not equal "
fi

if [ -z "$str1" ]
then echo "string 1 is null "
fi

if [ -z "$str2" ]
then echo "string 2 is null "
fi
```

Output

```
205120081@ca: ~
205120081@ca:~$ sh ques9.sh
 enter string 1 : ratan
 enter string 2 : nitt
 strings are not equal
205120081@ca:~$
```

Question 10 : Write a shell script to accept one integer argument and print its multiplication table.

Answer

```
#!/bin/bash
echo "Enter Number "
read n
i=1
while [ $i -le 10 ]
do
        echo " $n x $i = `expr $n \* $i`"
        i=`expr $i + 1`
done
```

Output

```
205120081@ca: ~
205120081@ca:~$ sh ques10.sh
Enter Number
5
 5 x 1 = 5
 5 x 2 = 10
 5 x 3 = 15
 5 x 4 = 20
 5 x 5 = 25
 5 x 6 = 30
 5 x 7 = 35
 5 x 8 = 40
 5 x 9 = 45
 5 x 10 = 50
205120081@ca:~$
```

Question 11 : Write a shell script, which accepts any number of arguments and prints them in the Reverse order. (For example, if the script is passed A B C as arguments, then execution should produce C B A on the standard output).

Answer

```
a=$#
echo "Number of arguments are" $a
x=$*
c=$a
res=''
while [ 1 -le $c ]
do
c=`expr $c - 1`
shift $c
res=$res' '$1
set $x
done
echo Arguments in reverse order $res
```

Output

205120081@ca: ~

```
205120081@ca:~$ sh ques11.sh ratan nit trichy
Number of arguments are 3
Arguments in reverse order trichy nit ratan
205120081@ca:~$
```

Question 12 : Write a Shell Script that makes use of grep to isolate the line in /etc/passwd that contains your login detailsu

username=’whoami’

grep “`whoami`” /etc/passwd

205120081@ca: ~

```
205120081@ca:~$ sh ques12.sh
205120081:x:1085:1085::/home/Students/MCA_2020_JULY_BATCH/205120081:/bin/bash
205120081@ca:~$
```

Question 13: Write a shell script to display all files in the /home/YourLoginName subdirectory as well as display the type of all files.

Answer

```
for file in *
do
if [ -d $file ]
then echo "$file is a dir "
elif [ -f $file ]
then echo "$file is a file "
elif [ -p $file ]
then echo "$file is a pipe "
else  echo "$file is a hyperlink "
fi
done
```

Output

205120081@ca: ~

```
205120081@ca:~$ vi ques13.sh
205120081@ca:~$ sh ques13.sh
a.out is a file
data is a file
file.txt is a file
listforques21 is a file
newpro is a file
nitt.txt is a file
q17.c is a file
queq17.c is a file
ques0.sh is a file
ques10.sh is a file
ques11.sh is a file
ques12.sh is a file
ques13.sh is a file
ques14.sh is a file
ques15.sh is a file
ques16.sh is a file
ques17.c is a file
ques18.c is a file
ques19.c is a file
ques2.sh is a file
ques20.c is a file
ques21.c is a file
ques22.c is a file
ques22_b.c is a file
ques23.c is a file
ques24.c is a file
ques25.c is a file
ques25_r.c is a file
ques25_w.c is a file
ques26.c is a file
ques27.c is a file
ques28_fcfs.c is a file
ques28_priority.c is a file
ques28_rr.c is a file
ques28_sjf.c is a file
ques29.c is a file
ques3.sh is a file
ques30_binarySemaphore.c is a file
```

```
ques30_counting.c is a file
ques31.c is a file
ques31_consumer.c is a file
ques32_reader.c is a file
ques32_writer.c is a file
ques33_bankers.c is a file
ques33_dining.c is a file
ques34_best.c is a file
ques34_first.c is a file
ques34_worst.c is a file
ques35.c is a file
ques4.sh is a file
ques5.sh is a file
ques6.sh is a file
ques7.sh is a file
ques8.sh is a file
ques9.sh is a file
tmp is a file
205120081@ca:~$
```

Question14 : Using shell script, display the contents of the present working directory. If it is an ordinary file print its permission and change the permissions to r--r--r--.

Answer

```
# !/bin/bash

for item in *
do
   if [ -f $item ]
   then
      echo "----------------$item----------------"
      if [ -x $item ]
      then
         echo "File in Executable mode"
         chmod -x $item
         echo "Executable permission Removed!"
      fi
      if [ -w $item ]
      then
         echo "File in Write mode"
         chmod -w $item
         echo "Write permission Removed!"
      fi
   if [ -r $item ]
   then
      echo "Already in read mode(r--r--r--)"
   else
      chmod +r $item
      echo "Now the read permission granted "
   fi
      echo "final permission"
      ls -al $item
   fi
   echo
done
```

Output

205120081@ca: ~

```
Executable permission Removed!
File in Write mode
Write permission Removed!
Already in read mode(r--r--r--)
final permission
-r-------- 1 205120081 205120081 796 Jun 25 16:30 ques4.sh

----------------ques5.sh----------------
File in Executable mode
Executable permission Removed!
File in Write mode
Write permission Removed!
Already in read mode(r--r--r--)
final permission
-r-------- 1 205120081 205120081 478 Jun 25 16:50 ques5.sh

----------------ques6.sh----------------
File in Executable mode
Executable permission Removed!
File in Write mode
Write permission Removed!
Already in read mode(r--r--r--)
final permission
-r-------- 1 205120081 205120081 168 Jun 25 16:51 ques6.sh

----------------ques8.sh----------------
File in Executable mode
Executable permission Removed!
File in Write mode
Write permission Removed!
Already in read mode(r--r--r--)
final permission
-r-------- 1 205120081 205120081 311 Jun 25 17:12 ques8.sh

----------------ques9.sh----------------
File in Executable mode
Executable permission Removed!
File in Write mode
Write permission Removed!
Already in read mode(r--r--r--)
final permission
-r-------- 1 205120081 205120081 277 Jun 25 16:52 ques9.sh

205120081@ca:~$
```

Question 15: Use find, grep and sort to display a sorted list of all files in the /home/YourLoginName subdirectory that contains the word “hello” somewhere inside them.

Answer find -type f -print0 | xargs -0 grep -li "hello" | sort

```
205120081@ca:~$ vi ques15.sh
205120081@ca:~$ sh ques15.sh
./ques0.sh
./ques15.sh
205120081@ca:~$
```

Question 16: Write a shell script to produce a list of users and their login shells.
Answer
finger $USER | grep 'Shell:*' | cut -f3 -d ":"

```
205120081@ca:~$ sh ques16.sh
ques16.sh: 1: finger: not found
205120081@ca:~$
```

CYCLE SHEET – II

Program 17 Write a C program to kill a process by specifying its name rather than its PID.
Source Code

```
#include<stdio.h>
#include<string.h>
Int main()
{
 char cmd[50],cmd1[50],cmd2[50],log[50],pname[50],pid[50];
 FILE * fp;
system("rm newpro");
system("rm data");
printf("enter ur login name\n");
fgets(log,sizeof(log),stdin);
strcpy(cmd,"ps -aux | grep ");
strcat(cmd,log);
system(cmd);
printf("enter the name of the process u want to terminate\n");
scanf("%s",pname);
strcpy(cmd1,"ps -a | grep ");
strcat(cmd1,pname);
strcat(cmd1," > newpro");
system(cmd1);
system("cut -f2 -d' ' newpro > data");
fp=fopen("data","r");
fscanf(fp,"%s",pid);
strcpy(cmd2,"kill ");
strcat(cmd2,pid);
system(cmd2);
system(cmd);
printf("the process %s is killed successfully",pname);
}
```

Output

```
205120081@ca:~$ ./a.out
enter ur login name
205120081
root      973552  0.0  0.0  13956  9040 ?        Ss   14:00   0:00 sshd: 205120081 [priv]
2051200+  973884  0.0  0.0  13956  5084 ?        S    14:00   0:00 sshd: 205120081@pts/3
2051200+  981443  0.0  0.0   2608   536 pts/3    S+   14:05   0:00 sh -c ps -aux | grep 205120081
2051200+  981445  0.0  0.0   6432   736 pts/3    S+   14:05   0:00 grep 205120081
enter the name of the process u want to terminate
root
sh: 1: kill: Illegal number: }
root      973552  0.0  0.0  13956  9040 ?        Ss   14:00   0:00 sshd: 205120081 [priv]
2051200+  973884  0.0  0.0  13956  5084 ?        S    14:00   0:00 sshd: 205120081@pts/3
2051200+  981616  0.0  0.0   2608   608 pts/3    S+   14:05   0:00 sh -c ps -aux | grep 205120081
2051200+  981618  0.0  0.0   6432  2560 pts/3    S+   14:05   0:00 grep 205120081
the process root is killed successfully205120081@ca:~$
```

Program 18 : Create a file with few lines, Write a C program to read the file and delete the spaces more than one in the file (use UNIX file API's).

Source code

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

int main()
{
	FILE * pfile;
	int a;
	printf("\n Remove the spaces between two words  :\n");
   printf("----------------------------------------\n");
// file.txt contain : the quick brown fox jumps over the lazy dog
	pfile=fopen ("file.txt","r");
	printf(" The content of the file is :\n The quick brown fox jumps over the lazy dog\n\n");
	printf(" After removing the spaces the content is : \n");
	if (pfile)
	{
		do {
			a = fgetc (pfile);
			if ( isgraph(a) ) putchar (a);
		} while (a != EOF);
		fclose (pfile);
	}
	printf("\n\n");
	return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques18.c
205120081@ca:~$ ./a.out

 Remove the spaces between two words  :
----------------------------------------
 The content of the file is :
 The quick brown fox jumps over the lazy dog

 After removing the spaces the content is :
Thequickbrownfoxjumpsoverthelazydog

205120081@ca:~$
```

Program 19 : Implement a C program to list the users who have logged in more than once.

Source code

```
#include<stdio.h>
#include<sys/utsname.h>
#include<utmp.h>

int main(void)
{
 struct utmp *n;
 char *a;
 int i;
 setutent();
 n=getutent();

 while(n!=NULL)
 {
  if(n->ut_type==7)
  {
   printf("%9s",n->ut_user);
   printf("%12s",n->ut_line);

   printf(" ");
   for(i=4;i<16;i++)
   printf("%c",a[i]);
   printf(" (");
   printf("%s10",n->ut_host);
   printf(")");
   printf("\n");
  }
  n=getutent();
 }
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques19.c
205120081@ca:~$ ./a.out
205120061       pts/0 (157.34.9.237)
205120041       pts/1 (182.69.98.125)
205120095       pts/2 (157.42.238.44)
205120081       pts/3 (27.61.64.96)
205120035       pts/4 (59.95.153.102)
205120087       pts/5 (114.134.24.61)
205120059       pts/6 (49.35.177.107)
205120021       pts/7 (103.39.116.183)
205120101       pts/8 (171.49.136.127)
205120079       pts/9 (49.36.47.104)
205120097      pts/10 (171.60.183.158)
205120105      pts/11 (47.9.209.186)
205120011      pts/12 (1.23.122.87)
205120055      pts/13 (171.79.101.102)
205120091      pts/14 (115.99.57.163)
205120053      pts/15 (223.236.59.97)
205120081      pts/16 (157.42.95.235)
205120025      pts/17 (157.42.238.28)
205120009      pts/18 (106.222.172.156)
205120043      pts/19 (106.76.156.65)
205120107      pts/20 (106.79.237.156)
205120093      pts/21 (139.167.196.144)
205120015      pts/22 (106.207.71.126)
205120015      pts/24 (106.207.71.126)
205120081@ca:~$
```

Program 20 : Write a C program which renames all .txt files as .text files

Source code

```
#include<stdio.h>

int main()
{
   // Old file name
   char old_name[] = "ratan.txt";

   // Any string
   char new_name[] = "nitt.txt";
   int value;

   // File name is changed here
   value = rename(old_name, new_name);

   // Print the result
   if(!value)
   {
      printf("%s", "File name changed successfully");
   }
   else
   {
      perror("Error");
   }
   return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques20.c
205120081@ca:~$ ./a.out
File name changed successfully205120081@ca:~$
```

Program 21 : Implement a C program that reports the number of file names in the current working directory that consist of exactly five characters.

Source code

```
#include <stdio.h>
#include <string.h>
#include<stdlib.h>
int main()
{
   FILE *fp;
   int i;
   char temp[50], cmd[20];
   system("dir > listforques21");
   fp = fopen("listforques21", "r");
   while (!feof(fp))
   {
      fscanf(fp, "%s", temp);
      if (strlen(temp) == 5)
      {
         printf(" %s \n", temp);
      }
   }
   return 0;
}
```

Output

```
205120081@ca:~$ vi ques21.c
205120081@ca:~$ gcc ques21.c
205120081@ca:~$ ./a.out
 a.out
 ql7.c
205120081@ca:~$
```

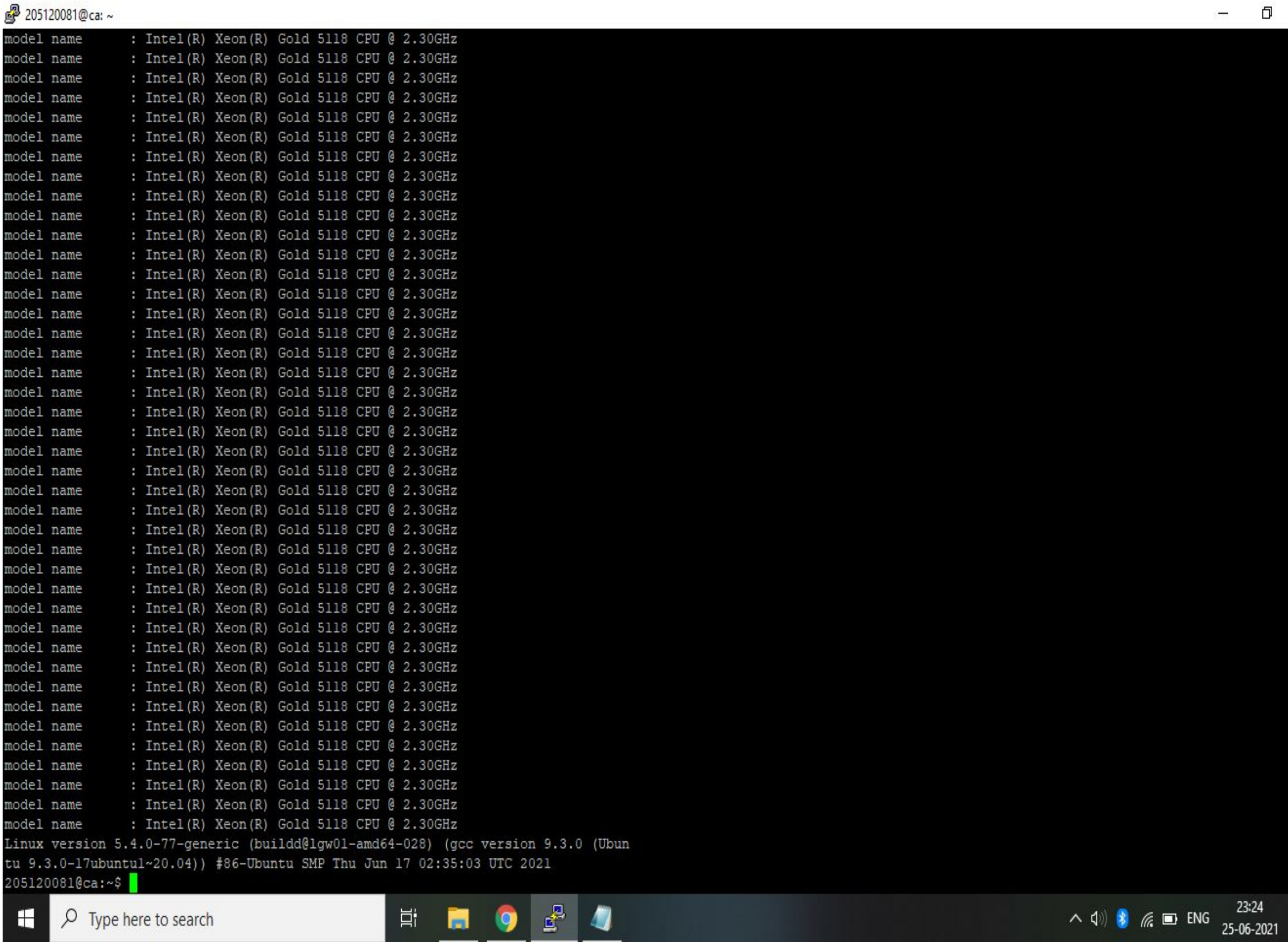
Program 22 : Write Programs to

22 a) Report the behaviour of the OS to get the CPU type and model, kernel version.

Source code

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 system("cat /proc/cpuinfo | grep model\\ name");
 system("cat /proc/version");
 return 0;
}
```

Output



```
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
model name      : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
Linux version 5.4.0-77-generic (buildd@lgw01-amd64-028) (gcc version 9.3.0 (Ubun
tu 9.3.0-17ubuntu1~20.04)) #86-Ubuntu SMP Thu Jun 17 02:35:03 UTC 2021
205120081@ca:~$
```

22 b) Get the amount of memory configured into the computer, amount of memory currently available.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   system("cat /proc/meminfo | grep Mem");
   return 0;
}
```

Output

```
205120081@ca:~$ vi ques22_b.c
205120081@ca:~$ gcc ques22_b.c
205120081@ca:~$ ./a.out
MemTotal:       131652288 kB
MemFree:        120295612 kB
MemAvailable:   126900716 kB
205120081@ca:~$
```

Program 23 Write a program to a create child process and display the process ID of parent and child processes.

Source Code

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{


   int pid = fork();

   if(pid)
   {
      printf("I am the child Process with %d pid\n", getpid());
   }
   else
   {
      printf("I am the Parent Process with %d pid\n", getpid());
   }

   return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques23.c
205120081@ca:~$ ./a.out
I am the child Process with 1546983 pid
I am the Parent Process with 1546984 pid
205120081@ca:~$
```

Program 24 Write a program to demonstrate the implementation of Inter Process Communication (IPC) "who | grep YourLoginName" using pipes.

Source Code

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
   int pid;
   int fd[2];
   pipe(fd);
   pid = fork();
   if (pid == -1)
   {
      perror("fork");
      exit(-1);
   }
   if (pid)
   {
      close(0);
      dup(fd[0]);
      close(fd[1]);
      execl("/usr/bin/wc", "wc", "-l", (char *)0);
```

```
    close(fd[0]);
  }
  else
  {
    close(1);
    dup(fd[1]);
    close(fd[0]);
    execl("/usr/bin/who", "who", (char *)0);
    close(fd[1]);
  }
  return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques24.c
205120081@ca:~$ ./a.out
Writing to pipe - Message from Process 1 is P1 started
Reading from pipe - Message from Process 1 is P1 started
Writing to pipe - Message from Process 2 is P2 Started
Reading from pipe - Message from Process 2 is P2 Started
205120081@ca:~$
```

Program 25 Write a program to demonstrate the implementation of Inter Process Communication (IPC) using Message Queues.

Source code

Writer

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define MAX 20

struct msg_buffer
{
  long msg_type;
  char msg_text[100];
} message;

int main()
{
  key_t key;
  int msgid;

  key = ftok("process", 110);

  msgid = msgget(key, 0666 | IPC_CREAT);
  message.msg_type = 1;

  printf("Write Data : ");
  fgets(message.msg_text, MAX, stdin);
  msgsnd(msgid, &message, sizeof(message), 0);
  printf("Message queued : %s \n", message.msg_text);
```

```
    return 0;
}
```

output

```
205120081@ca: ~
205120081@ca:~$ gcc -lpthread ques25_w.c
205120081@ca:~$ ./a.out
Write Data : Hello Nitian
Message queued : Hello Nitian

205120081@ca:~$
```

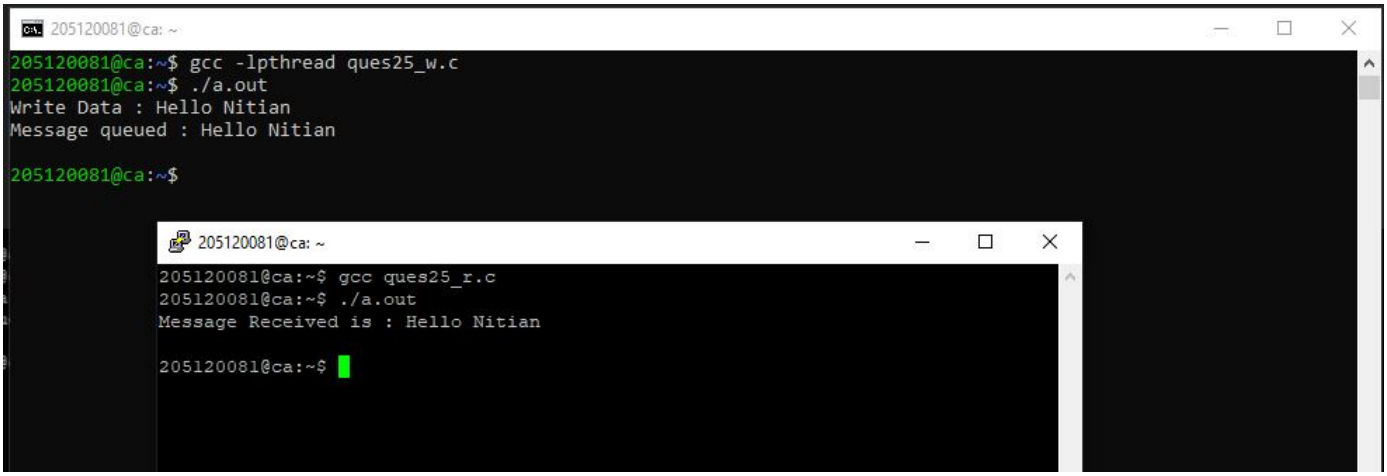
<u>Reader</u>

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct msg_buffer
{
    long msg_type;
    char msg_text[100];
} message;

int main()
{
    key_t key;
    int msgid;

    key = ftok("process", 110);

    msgid = msgget(key, 0666 | IPC_CREAT);
    msgrcv(msgid, &message, sizeof(message), 1, 0);

    printf("Message Received is : %s \n", message.msg_text);
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques25_r.c
205120081@ca:~$ ./a.out
Message Received is : Hello Nitian

205120081@ca:~$
```

Output 25



Program 26 : Write a program to demonstrate the implementation of Inter Process Communication (IPC) using shared memory.

Source code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>
#define SEGSIZE 100
int main(int argc, char *argv[])
{
   int shmid, cntr;
   key_t key;
   char *segptr;
   char buff[] = "poooda.....";
   key = ftok(".", 's');
   if ((shmid = shmget(key, SEGSIZE, IPC_CREAT | IPC_EXCL | 0666)) == -1)
   {
      if ((shmid = shmget(key, SEGSIZE, 0)) == -1)
      {
         perror("shmget");
         exit(1);
      }
   }
   else
   {
      printf("Creating a new shared memory seg \n");
      printf("SHMID:%d", shmid);
   }
   system("ipcs –m");
   if ((segptr = (char *)shmat(shmid, 0, 0)) == (char *)-1)
   {
      perror("shmat");
      exit(1);
   }
   printf("Writing data to shared memory...\n");
   strcpy(segptr, buff);
```

```
    printf("DONE\n");
    printf("Reading data from shared memory...\n");
    printf("DATA:-%s\n", segptr);
    printf("DONE\n");
    printf("Removing shared memory Segment...\n");
    if (shmctl(shmid, IPC_RMID, 0) == -1)
        printf("Can‟t Remove Shared memory Segment...\n");
    else
        printf("Removed Successfully");
}
```

205120081@ca: ~

```
205120081@ca:~$ gcc ques26.c
205120081@ca:~$ ./a.out
Creating a new shared memory seg

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x000004d2 3          205120077  666        0            0
0xffffffff 10         205120109  666        112          1

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 196610     super      600        16384      1          dest
0x00000000 196616     super      600        524288     2          dest
0x00000000 196621     super      600        524288     2          dest
0x00001388 262160     205120087  222        27         0
0x00000000 196628     super      600        524288     2          dest
0x00000000 196629     super      600        524288     2          dest
0x00000000 196630     super      600        524288     2          dest
0x00000000 196631     super      600        524288     2          dest
0x73000453 262168     205120081  666        100        0
0x00000000 196654     lightdm    600        67108864   2          dest
0x00001f40 196655     205120113  666        27         0
0x000004f5 196656     205120007  666        27         0
0x00000000 196658     super      600        5242880    2          dest
0x00000000 229430     super      600        524288     2          dest
0x000004d2 196663     205120033  666        27         0
0x0000162e 196668     205120061  666        27         0
0x00000929 196669     205120093  666        1024       0

------ Semaphore Arrays --------
key        semid      owner      perms      nsems
0x00000500 1          205120113  666        1
0x00000000 3          daemon     600        1
0x00000000 5          daemon     600        1
0x00000000 6          daemon     600        1
0x00000000 7          daemon     600        1
0x00000000 8          daemon     600        1
0x00000000 9          daemon     600        1
```

```
SHMID:262168Writing data to shared memory...
DONE
Reading data from shared memory...
DATA:-poooda......
DONE
Removing shared memory Segment...
Removed Successfully205120081@ca:~$
```

Program 27 : Write a program to create a thread and let the thread check whether the given number is prime or not.

Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include <unistd.h>
#define MAX_THREAD
int n;
void *isPrime(void *vargp)
{
    if(n%2)
        printf("Odd Number\n");
```

```
    else
        printf("Even Number\n");
    int flag = 0;
    for(int i=2; i<n; i++)
    {
        if(n%i==0)
        {
            flag=1;
            break;
        }
    }
    if(flag)
        printf("Not Prime\n");
    else
        printf("Prime\n");
}
int main()
{
    printf("Enter the number :");
    scanf("%d",&n);
    pthread_t thread_id;
    printf("Thread Created\n");
    pthread_create(&thread_id, NULL, isPrime, NULL);
    (void)pthread_join(thread_id, NULL);
    printf("Thread Joined\n");
    exit(0);
    return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc -pthread ques27.c
205120081@ca:~$ ./a.out
Enter the number :13
Thread Created
Odd Number
Prime
Thread Joined
205120081@ca:~$
```

Program 28 Implement FCFS, SJF, Priority and Round– Robin process scheduling algorithms.
Source Code

FCFS

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

int max(int a, int b)
{
    if(a<b)
        return b;
    return a;
```

```
}
struct Process {
   int arrival;
   int burst;
   int prioity;
};

int main(){

   printf("Simulating First-come First-serve Process Scheduling Algorithm");
   printf("\n Enter number of process: ");
   int n;
   scanf("%d",&n);
   struct Process *arr = (struct Process *)malloc(sizeof(struct Process)*n);
   for(int i=0; i<n; i++)
   {
      printf("Enter Arrival time for process #%d: ",i+1);
      scanf("%d",&arr[i].arrival);
   }
   for(int i=0; i<n; i++)
   {
      printf("Enter Burst time for process #%d: ",i+1);
      scanf("%d",&arr[i].burst);
   }
   for(int i=0; i<n; i++)
   {
      printf("Enter Priority time for process #%d: ",i+1);
      scanf("%d",&arr[i].prioity);
   }
   /* Waiting Time Caluclation */
   int *wt = (int *)malloc(sizeof(int)*n);
   wt[0] = 0;
   double tot_wt=0;
   for(int i=1; i<n; i++)
   {
      wt[i] = arr[i-1].burst+wt[i-1]-(arr[i].arrival-arr[i-1].arrival);
      tot_wt += wt[i];
   }

   int *tat = (int *)malloc(sizeof(int)*n);
   double tot_tat = 0;
   for(int i=0; i<n; i++)
   {
      tat[i] = wt[i]+arr[i].burst;
      tot_tat += tat[i];
   }
   double count = n;
   double avg_wt, avg_tat;
   avg_wt = tot_wt/count;
   avg_tat = tot_tat/count;

   printf("\n");
   printf("Process\t\tArrival\t\tBurst\t\tPriority\tWait\t\tTat\n");
   for (int i = 0; i < n; i++)
```

```
    {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i+1,arr[i].arrival,arr[i].burst,arr[i].prioity,wt[i],tat[i]);
    }

printf("\n_____________________________________________________________________________________________\n
");
    printf("\t\t\t\t\t\t\t\t%f\t\t%f\t\n",tot_wt,tot_tat);
    printf("Average Waiting Time : %f\n",avg_wt);
    printf("Average Turnaround Time : %f\n",avg_tat);


    return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques28_fcfs.c
205120081@ca:~$ ./a.out
Simulating First-come First-serve Process Scheduling Algorithm
 Enter number of process: 4
Enter Arrival time for process #1: 3
Enter Arrival time for process #2: 2
Enter Arrival time for process #3: 6
Enter Arrival time for process #4: 9
Enter Burst time for process #1: 1
Enter Burst time for process #2: 2
Enter Burst time for process #3: 6
Enter Burst time for process #4: 5
Enter Priority time for process #1: 4
Enter Priority time for process #2: 3
Enter Priority time for process #3: 8
Enter Priority time for process #4: 6

Process         Arrival         Burst           Priority        Wait            T
at
1               3               1               4               0               1
2               2               2               3               2               4
3               6               6               8               0               6
4               9               5               6               3               8


_____________________________________________________________________________
_______
                                                                5.000000        1
9.000000
Average Waiting Time : 1.250000
Average Turnaround Time : 4.750000
205120081@ca:~$
```

SJF

```
#include<stdio.h>
 int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);

    printf("nEnter Burst Time:n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }
```

```
  //sorting of burst times
   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(bt[j]<bt[pos])
             pos=j;
      }

      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;

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

   wt[0]=0;


   for(i=1;i<n;i++)
   {
      wt[i]=0;
      for(j=0;j<i;j++)
         wt[i]+=bt[j];

      total+=wt[i];
   }

   avg_wt=(float)total/n;
   total=0;

   printf("nProcesst    Burst Time    tWaiting TimetTurnaround Time");
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];
      total+=tat[i];
      printf("np%dtt  %dtt    %dttt%d",p[i],bt[i],wt[i],tat[i]);
   }

   avg_tat=(float)total/n;
   printf("nnAverage Waiting Time=%f",avg_wt);
   printf("nAverage Turnaround Time=%fn",avg_tat);
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ vi ques28_sjf.c
205120081@ca:~$ gcc ques28_sjf.c
205120081@ca:~$ ./a.out
Enter number of process:4
nEnter Burst Time:np1:3
p2:1
p3:2
p4:6
nProcesst    Burst Time    tWaiting TimetTurnaround Timenp2tt  1tt    0ttt1np3tt
  2tt    1ttt3np1tt  3tt    3ttt6np4tt  6tt    6ttt12nnAverage Waiting Time=2.50
205120081@ca:~$
```

Priority

```
#include <stdio.h>
int main()
{

    int bt[20],wt[20],p[20],tat[20],priority[20];
    float avwt=0,avtat=0;

    int i,j,n,temp,key;

    printf("\nEnter the number of the processes: ");

    scanf("%d",&n);

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

    {

      printf("\nEnter the burst time and priority of the process P[%d]: ",i);

      scanf("%d",&bt[i]);
      scanf("%d",&priority[i]);
      p[i]=i;

    }

    for(i=0;i<n;i++)
    {
      key=i;
      for(j=i+1;j<n;j++)
      {
        if(priority[j]<priority[key])
        {
          key=j;
        }
      }
      temp=bt[i];
      bt[i]=bt[key];
      bt[key]=temp;

      temp=priority[i];
      priority[i]=priority[key];
```

```
        priority[key]=temp;

        temp=p[i];
        p[i]=p[key];
        p[key]=temp;
    }


    wt[0]=0;
    tat[0]=bt[0];
    avtat=tat[0];

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

        wt[i]=wt[i-1]+bt[i-1];

        tat[i]=tat[i-1]+bt[i];

        avwt+=wt[i];

        avtat+=tat[i];

    }

    avwt=avwt/n;
    avtat=avtat/n;

    printf("\n\nPROCESS\t\twaiting time\tburst time\tTurnaround time\n");

    printf("\n");

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

        printf("P[%d]\t\t%d\t\t%d\t\t%d\n",p[i],wt[i],bt[i],tat[i]);

    }

    printf("\n\nAverage waiting time: %.2f",avwt);
    printf("\n\nAverage Turn around time is: %.2f",avtat);
    printf("\n");

    return 0;

}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques28_priority.c
205120081@ca:~$ ./a.out

Enter the number of the processes: 4

Enter the burst time and priority of the process P[0]: 5
9

Enter the burst time and priority of the process P[1]: 2
1

Enter the burst time and priority of the process P[2]: 3
6

Enter the burst time and priority of the process P[3]: 7
8


PROCESS         waiting time    burst time      Turnaround time

P[1]            0               2               2
P[2]            2               3               5
P[3]            5               7               12
P[0]            12              5               17


Average waiting time: 4.75

Average Turn around time is: 9.00
205120081@ca:~$
```

Round– Robin

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

void main()
{
   // initlialize the variable name
   int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
   float avg_wt, avg_tat;
   printf(" Total number of process in the system: ");
   scanf("%d", &NOP);
   y = NOP; // Assign the number of process to variable y

// Use for loop to enter the details of the process like Arrival time and the Burst Time
for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");  // Accept arrival time
scanf("%d", &at[i]);
printf(" \nBurst time is: \t"); // Accept the Burst time
scanf("%d", &bt[i]);
temp[i] = bt[i]; // store the burst time in temp array
}
// Accept the Time qunat
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
// Display the process No, burst time, Turn Around Time and the waiting time
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
```

```
{
if(temp[i] <= quant && temp[i] > 0) // define the conditions
{
   sum = sum + temp[i];
   temp[i] = 0;
   count=1;
   }
   else if(temp[i] > 0)
   {
      temp[i] = temp[i] - quant;
      sum = sum + quant;
   }
   if(temp[i]==0 && count==1)
   {
      y--; //decrement the process no.
      printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
      wt = wt+sum-at[i]-bt[i];
      tat = tat+sum-at[i];
      count =0;
   }
   if(i==NOP-1)
   {
      i=0;
   }
   else if(at[i+1]<=sum)
   {
      i++;
   }
   else
   {
      i=0;
   }
}
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
getch();
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques28_rr.c
205120081@ca:~$ ./a.out
 Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       9

Burst time is:  4

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       2

Burst time is:  3

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       6

Burst time is:  8

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       9

Burst time is:  6
Enter the Time Quantum for the process:         3

 Process No             Burst Time             TAT            Waiting Time
Process No[2]           3                              4                      1
Process No[1]           4                              4                      0
Process No[4]           6                              10                     4
Process No[3]           8                              15                     7
 Average Turn Around Time:      3.000000
 Average Waiting Time:  8.250000205120081@ca:~$
```

Program 29 Write a program to perform a tidy exit on receipt of an interrupt signal.

Source Code

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

FILE *temp_file;
void leave(int sig);

int main()
{
   (void)signal(SIGINT, leave);
   temp_file = fopen("tmp", "w");
   for (;;)
   {
      /*
            * Do things....
            */
      printf("Ready...\n");
      (void)getchar();
   }
   /* can't get here ... */
   exit(EXIT_SUCCESS);
   return 0;
}

// SIGHUP      1      Hang up detected on controlling terminal or death of con
trolling process
// SIGINT      2      Issued if the user sends an interrupt signal (Ctrl + C)
```

```
// SIGQUIT     3      Issued if the user sends a quit signal (Ctrl + D)
// SIGFPE      8      Issued if an illegal mathematical operation is attempted
// SIGKILL     9      If a process gets this signal it must quit immediately a
nd will not perform any clean-up operations
// SIGALRM     14     Alarm clock signal (used for timers)
// SIGTERM     15     Software termination signal (sent by kill by default)
// kill -l
void leave(int sig)
{
   printf("\nSIGINT Recieved, Exiting");
   fprintf(temp_file, "\nInterrupted..\n");
   fclose(temp_file);
   exit(sig);
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques29.c
205120081@ca:~$ ./a.out
Ready...

Ready...

Ready...

Ready...

Ready...

Ready...

Ready...
```

Program 30 Implement a) Binary Semaphore b) Counting Semaphore

Source Code a) Binary Semaphore

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

sem_t mutex;

void* thread(void* arg)
{
      //wait
      sem_wait(&mutex);
      printf("\nEntered..\n");

      //critical section
      sleep(4);

      //signal
      printf("\nJust Exiting...\n");
      sem_post(&mutex);
}
```

```
int main()
{
      sem_init(&mutex, 0, 1);
      pthread_t t1,t2;
      pthread_create(&t1,NULL,thread,NULL);
      sleep(2);
      pthread_create(&t2,NULL,thread,NULL);
      pthread_join(t1,NULL);
      pthread_join(t2,NULL);
      sem_destroy(&mutex);
      return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques30_binarySemaphore.c -lpthread -lrt
205120081@ca:~$ ./a.out

Entered..

Just Exiting...

Entered..

Just Exiting...
205120081@ca:~$
```

Program 31 : Write a program to demonstrate the implementation of Producer and Consumer problem.

Source Code

```
#include <stdio.h>
int main()
{
 int buffer[10], bufsize, in, out, produce, consume, choice = 0;
 in = 0;
 out = 0;
 bufsize = 10;
 while (choice != 3)
 {
  printf("\n 1.Produce\t 2.Consume\t 3.Exit");
  printf("\nEnter your choice: ");
  scanf("%d", &choice);
  switch (choice)
  {
  case 1:
   if ((in + 1) % bufsize == out)
    printf("\nBuffer is Full");
   else
   {
    printf("\nEnter the value: ");
    scanf("%d", &produce);
    buffer[in] = produce;
    in = (in + 1) % bufsize;
   }
   break;
```

```
    case 2:
     if (in == out)
       printf("\nBuffer is Empty\n");
     else
     {
       consume = buffer[out];
       printf("\nThe consumed value is %d\n", consume);
       out = (out + 1) % bufsize;
     }
     break;
    }
  }
  return 0;
}
```

Output

```
205120081@ca:~$ gcc ques31.c
205120081@ca:~$ ./a.out

 1.Produce       2.Consume       3.Exit
Enter your choice: 1

Enter the value: 2

 1.Produce       2.Consume       3.Exit
Enter your choice: 2

The consumed value is 2

 1.Produce       2.Consume       3.Exit
Enter your choice: 3
205120081@ca:~$
```

Program 32 : Write a program to implement Reader – Writer's problem

Source Code

ques32_writer.c

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
// structure for message queue
struct msg_buffer {
   long msg_type;
   char msg[100];
} message;

int main() {
   key_t my_key;
   int msg_id;
   my_key = ftok("progfile", 65); //create unique key
   msg_id = msgget(my_key, 0666 | IPC_CREAT); //create message queue and return id
   message.msg_type = 1;
   printf("Write Message : ");
   fgets(message.msg, 100, stdin);
   msgsnd(msg_id, &message, sizeof(message), 0); //send message
```

```
    printf("Sent message is : %s \n", message.msg);
}
```

Output

```
205120081@ca:~$ gcc -lpthread ques32_writer.c
205120081@ca:~$ ./a.out
Write Message : nit trichy
Sent message is : nit trichy

205120081@ca:~$
```

ques32_reader.c

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>
// Define message queue structure
struct msg_buffer {
    long msg_type;
    char msg[100];
} message;

int main() {
    key_t my_key;
    int msg_id;
    my_key = ftok("progfile", 65); //create unique key
    msg_id = msgget(my_key, 0666 | IPC_CREAT); //create message queue and return id
    msgrcv(msg_id, &message, sizeof(message), 1, 0); //used to receive message
    // display the message
    int n=strlen(message.msg);
    for(int i=0;i<n;i++)
    {
        if(message.msg[i]>='A'&&message.msg[i]<='Z')
        {
            message.msg[i]=message.msg[i]-'A'+'a';
        }
    else if(message.msg[i]>='a'&&message.msg[i]<='z')
        {
            message.msg[i]=message.msg[i]-'a'+'A';
        }
    }
    printf("Received Message is : %s \n", message.msg);
    msgctl(msg_id, IPC_RMID, NULL); //destroy the message queue
    return 0;
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ 205120081@ca:~$ gcc -lpthread ques32_reader.c
205120081@ca:~$ ./a.out
Received Message is : NIT TRICHY

205120081@ca:~$
```

Output 32

```
205120081@ca:~$ gcc -lpthread ques32_writer.c
205120081@ca:~$ ./a.out
Write Message : nit trichy
Sent message is : nit trichy

205120081@ca:~$
```

```
205120081@ca: ~
205120081@ca:~$ 205120081@ca:~$ gcc -lpthread ques32_reader.c
205120081@ca:~$ ./a.out
Received Message is : NIT TRICHY

205120081@ca:~$
```

Program 33 Write a program to implement Dining Philosopher's problem. Implement Banker's algorithm.

ques33_dining.c

Source Code

```
#include<stdio.h>

#define n 4

int compltedPhilo = 0,i;

struct fork{
int taken;
}ForkAvil[n];

struct philosp{
int left;
int right;
}Philostatus[n];

void goForDinner(int philID){ //same like threads concept here cases implemented
if(Philostatus[philID].left==10 && Philostatus[philID].right==10)
      printf("Philosopher %d completed his dinner\n",philID+1);
//if already completed dinner
else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){
         //if just taken two forks
         printf("Philosopher %d completed his dinner\n",philID+1);

         Philostatus[philID].left = Philostatus[philID].right = 10; //remembering that he completed dinner by assigning
value 10
         int otherFork = philID-1;

         if(otherFork== -1)
            otherFork=(n-1);

         ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0; //releasing forks
         printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);
         compltedPhilo++;
      }
      else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){ //left already taken, trying for right fork
            if(philID==(n-1)){
```

```
            if(ForkAvil[philID].taken==0){ //KEY POINT OF THIS PROBLEM, THAT LAST PHILOSOPHER TRYING IN reverse
DIRECTION
              ForkAvil[philID].taken = Philostatus[philID].right = 1;
              printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);
            }else{
              printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);
            }
          }else{ //except last philosopher case
            int dupphilID = philID;
            philID-=1;

            if(philID== -1)
              philID=(n-1);

            if(ForkAvil[philID].taken == 0){
              ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
              printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);
            }else{
              printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);
            }
          }
        }
        else if(Philostatus[philID].left==0){ //nothing taken yet
            if(philID==(n-1)){
              if(ForkAvil[philID-1].taken==0){ //KEY POINT OF THIS PROBLEM, THAT LAST PHILOSOPHER TRYING IN
reverse DIRECTION
                ForkAvil[philID-1].taken = Philostatus[philID].left = 1;
                printf("Fork %d taken by philosopher %d\n",philID,philID+1);
              }else{
                printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);
              }
            }else{ //except last philosopher case
              if(ForkAvil[philID].taken == 0){
                ForkAvil[philID].taken = Philostatus[philID].left = 1;
                printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);
              }else{
                printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1);
              }
            }
      }else{}
}

int main(){
for(i=0;i<n;i++)
      ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;

while(compltedPhilo<n){
/* Observe here carefully, while loop will run until all philosophers complete dinner
Actually problem of deadlock occur only thy try to take at same time
This for loop will say that they are trying at same time. And remaining status will print by go for dinner function
*/
for(i=0;i<n;i++)
        goForDinner(i);
printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);
```

```
}

return 0;
}
```

Output

```
205120081@ca:~$ gcc ques33_dining.c
205120081@ca:~$ ./a.out
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Fork 4 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 4
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 2

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Fork 4 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3

Till now num of philosophers completed dinner are 4

205120081@ca:~$
```

ques33_bankers.c

```
#include <stdio.h>

int current[5][5], maximum_claim[5][5], available[5];
int allocation[5] = {0, 0, 0, 0, 0};
int maxres[5], running[5], safe = 0;
int counter = 0, i, j, exec, resources, processes, k = 1;

int main()
{
printf("\nEnter number of processes: ");
    scanf("%d", &processes);

    for (i = 0; i < processes; i++)
{
        running[i] = 1;
        counter++;
    }

    printf("\nEnter number of resources: ");
    scanf("%d", &resources);

    printf("\nEnter Claim Vector:");
    for (i = 0; i < resources; i++)
{
       scanf("%d", &maxres[i]);
    }

   printf("\nEnter Allocated Resource Table:\n");
    for (i = 0; i < processes; i++)
{
       for(j = 0; j < resources; j++)
{
   scanf("%d", &current[i][j]);
        }
    }

    printf("\nEnter Maximum Claim Table:\n");
    for (i = 0; i < processes; i++)
{
        for(j = 0; j < resources; j++)
{
            scanf("%d", &maximum_claim[i][j]);
        }
    }

printf("\nThe Claim Vector is: ");
    for (i = 0; i < resources; i++)
{
       printf("\t%d", maxres[i]);
}

    printf("\nThe Allocated Resource Table:\n");
    for (i = 0; i < processes; i++)
```

```
{
      for (j = 0; j < resources; j++)
{
          printf("\t%d", current[i][j]);
       }
printf("\n");
    }

    printf("\nThe Maximum Claim Table:\n");
    for (i = 0; i < processes; i++)
{
       for (j = 0; j < resources; j++)
{
      printf("\t%d", maximum_claim[i][j]);
       }
       printf("\n");
    }

    for (i = 0; i < processes; i++)
{
       for (j = 0; j < resources; j++)
{
          allocation[j] += current[i][j];
       }
    }

    printf("\nAllocated resources:");
    for (i = 0; i < resources; i++)
{
       printf("\t%d", allocation[i]);
    }

    for (i = 0; i < resources; i++)
{
      available[i] = maxres[i] - allocation[i];
}

    printf("\nAvailable resources:");
    for (i = 0; i < resources; i++)
{
       printf("\t%d", available[i]);
    }
    printf("\n");

    while (counter != 0)
{
       safe = 0;
       for (i = 0; i < processes; i++)
{
          if (running[i])
{
             exec = 1;
             for (j = 0; j < resources; j++)
{
```

```
                if (maximum_claim[i][j] - current[i][j] > available[j])
{
                    exec = 0;
                    break;
                }
            }
            if (exec)
{
                printf("\nProcess%d is executing\n", i + 1);
                running[i] = 0;
                counter--;
                safe = 1;

                for (j = 0; j < resources; j++)
{
                    available[j] += current[i][j];
                }
            break;
            }
        }
    }
    if (!safe)
{
        printf("\nThe processes are in unsafe state.\n");
        break;
    }
else
{
        printf("\nThe process is in safe state");
        printf("\nAvailable vector:");

        for (i = 0; i < resources; i++)
{
            printf("\t%d", available[i]);
        }

    printf("\n");
    }
  }
  return 0;
}
```

```
The Claim Vector is:    4       5       6       9
The Allocated Resource Table:
        7       1       2       3
        7       8       9       6
        5       6       4       5
        3       6       7       10
        45      1       3       9

The Maximum Claim Table:
        5       1       4       6
        9       5       8       7
        5       2       6       3
        9       7       5       4
        6       32      4       5

Allocated resources:    67      22      25      33
Available resources:    -63     -17     -19     -24

The processes are in unsafe state.
205120081@ca:~$
```

Program 34 Implement the First Fit, Best Fit and Worst Fit file allocation strategy

Source Code

<u>First fit</u>

```
#include <stdio.h>
//#include <conio.h>
#define max 25
void main()
{
   int frag[max], b[max], f[max], i, j, nb, nf, temp;
   static int bf[max], ff[max];
  //  clrscr();
   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]);
  // getch();
}
```

205120081@ca: ~

```
205120081@ca:~$ gcc ques34_first.c
205120081@ca:~$ ./a.out

        Memory Management Scheme - First Fit
Enter the number of blocks:5
Enter the number of files:3

Enter the size of the blocks:-
Block 1:2
Block 2:6
Block 3:3
Block 4:9
Block 5:6
Enter the size of the files :-
File 1:7
File 2:5
File 3:2

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               7               4               9               2
2               5               2               6               1
3               2               1               2               0205120081@ca:~$
205120081@ca:~$
```

Best Fit

```
#include <stdio.h>
#include <conio.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];
   clrscr();
   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]);
    getch();
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ gcc ques34_best.c
205120081@ca:~$ ./a.out

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

Enter the size of the blocks:-
Block 1:10
Block 2:12
Block 3:36
Block 4:21
Block 5:63
Enter the size of the files :-
File 1:2
File 2:3
File 3:9
File 4:7

File No File Size       Block No        Block Size      Fragment
1               2               1               10              8
2               3               2               12              9
3               9               4               21              12
4               7               3               36              292051200 81@ca:~
205120081@ca:~$
```

34 Worst fit

```
#include <stdio.h>
#include <conio.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];
   clrscr();
   printf("\n\tMemory Management Scheme - Worst 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) //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]);
   getch();
}
```

Output

```
205120081@ca: ~
205120081@ca:~$ vi ques34_worst.c
205120081@ca:~$ gcc ques34_worst.c
205120081@ca:~$ ./a.out

        Memory Management Scheme - Worst Fit
Enter the number of blocks:5
Enter the number of files:6

Enter the size of the blocks:-
Block 1:2
Block 2:6
Block 3:9
Block 4:7
Block 5:5
Enter the size of the files :-
File 1:8
File 2:3
File 3:6
File 4:7
File 5:4
File 6:5

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               8               3               9               1
2               3               4               7               4
3               6               0               2496            0
4               7               0               2496            0
5               4               2               6               2
6               5               0               2496            0205120081@ca:~$
```

Program 35. Implement FIFO, Optimal, LRU and LFU page replacement algorithms.

Source Code

```
#include <stdio.h>
int n, nf;
int in[100];
int p[50];
int hit = 0;
int i, j, k;
int pgfaultcnt = 0;
void getData()
{
   printf("\nEnter length of page reference sequence:");
   scanf("%d", &n);
   printf("\nEnter the page reference sequence:");
   for (i = 0; i < n; i++)
   {
      scanf("%d", &in[i]);
   }
   printf("\nEnter no of frames:");
   scanf("%d", &nf);
}
void initialize()
{
   pgfaultcnt = 0;
   for (i = 0; i < nf; i++)
      p[i] = 9999;
}
int isHit(int data)
{
```

```
    hit = 0;
    for (j = 0; j < nf; j++)
    {
        if (p[j] == data)
        {
            hit = 1;
            break;
        }
    }
    return hit;
}
int getHitIndex(int data)
{
    int hitind;
    for (k = 0; k < nf; k++)
    {
        if (p[k] == data)
        {
            hitind = k;
            break;
        }
    }
    return hitind;
}
void dispPages()
{
    for (k = 0; k < nf; k++)
    {
        if (p[k] != 9999)
            printf(" %d", p[k]);
    }
}
void dispPgFaultCnt()
{
    printf("\nTotal no of page faults:%d", pgfaultcnt);
}
void fifo()
{
    initialize();
    for (i = 0; i < n; i++)
    {
        printf("\nFor %d :", in[i]);
        if (isHit(in[i]) == 0)
        {
            for (k = 0; k < nf - 1; k++)
                p[k] = p[k + 1];
            p[k] = in[i];
            pgfaultcnt++;
            dispPages();
        }
        else
            printf("No page fault");
    }
    dispPgFaultCnt();
```

```
}
void optimal()
{
   initialize();
   int near[50];
   for (i = 0; i < n; i++)
   {
      printf("\nFor %d :", in[i]);
      if (isHit(in[i]) == 0)
      {
         for (j = 0; j < nf; j++)
         {
            int pg = p[j];
            int found = 0;
            for (k = i; k < n; k++)
            {
               if (pg == in[k])
               {
                  near[j] = k;
                  found = 1;
                  break;
               }
               else
                  found = 0;
            }
            if (!found)
               near[j] = 9999;
         }
         int max = -9999;
         int repindex;
         for (j = 0; j < nf; j++)
         {
            if (near[j] > max)
            {
               max = near[j];
               repindex = j;
            }
         }
         p[repindex] = in[i];
         pgfaultcnt++;
         dispPages();
      }
      else
         printf("No page fault");
   }
   dispPgFaultCnt();
}
void lru()
{
   initialize();
   int least[50];
   for (i = 0; i < n; i++)
   {
      printf("\nFor %d :", in[i]);
```

```
        if (isHit(in[i]) == 0)
        {
            for (j = 0; j < nf; j++)
            {
                int pg = p[j];
                int found = 0;
                for (k = i - 1; k >= 0; k--)
                {
                    if (pg == in[k])
                    {
                        least[j] = k;
                        found = 1;
                        break;
                    }
                    else
                        found = 0;
                }
                if (!found)
                    least[j] = -9999;
            }
            int min = 9999;
            int repindex;
            for (j = 0; j < nf; j++)
            {
                if (least[j] < min)
                {
                    min = least[j];
                    repindex = j;
                }
            }
            p[repindex] = in[i];
            pgfaultcnt++;
            dispPages();
        }
        else
            printf("No page fault!");
    }
    dispPgFaultCnt();
}
void lfu()
{
    int usedcnt[100];
    int least, repin, sofarcnt = 0, bn;
    initialize();
    for (i = 0; i < nf; i++)
    usedcnt[i] = 0;
    for (i = 0; i < n; i++)
    {
        printf("\n For %d :", in[i]);
        if (isHit(in[i]))
        {
            int hitind = getHitIndex(in[i]);
            usedcnt[hitind]++;
            printf("No page fault!");
```

```
        }
        else
        {
            pgfaultcnt++;
            if (bn < nf)
            {
                p[bn] = in[i];
                usedcnt[bn] = usedcnt[bn] + 1;
                bn++;
            }
            else
            {
                least = 9999;
                for (k = 0; k < nf; k++)
                    if (usedcnt[k] < least)
                    {
                        least = usedcnt[k];
                        repin = k;
                }
                p[repin] = in[i];
                sofarcnt = 0;
                for (k = 0; k <= i; k++)
                    if (in[i] == in[k])
                        sofarcnt = sofarcnt + 1;
                        usedcnt[repin] = sofarcnt;
            }
            dispPages();
        }
    }
    dispPgFaultCnt();
}
void secondchance()
{
    int usedbit[50];
    int victimptr = 0;
    initialize();
    for (i = 0; i < nf; i++)
    usedbit[i] = 0;
    for (i = 0; i < n; i++)
    {
        printf("\nFor %d:", in[i]);
        if (isHit(in[i]))
        {
            printf("No page fault!");
            int hitindex = getHitIndex(in[i]);
            if (usedbit[hitindex] == 0)
                usedbit[hitindex] = 1;
        }
        else
        {
            pgfaultcnt++;
            if (usedbit[victimptr] == 1)
            {
                do
```

```
            {
                usedbit[victimptr] = 0;
                victimptr++;
                if (victimptr == nf)
                    victimptr = 0;
            }
            while (usedbit[victimptr] != 0);
        }
        if (usedbit[victimptr] == 0)
        {
            p[victimptr] = in[i];
            usedbit[victimptr] = 1;
            victimptr++;
        }
        dispPages();
    }
    if (victimptr == nf)
        victimptr = 0;
  }
  dispPgFaultCnt();
}
int main()
{
  int choice;
  while (1)
  {
    printf("\nPage Replacement Algorithms\n1.Enter data\n2.FIFO\n3.Optimal\n4.LRU\n5.LFU\n6.Second
Chance\n7.Exit\nEnter your choice:");
    scanf("%d", &choice);
    switch (choice)
    {
        case 1 :
            getData();
            break;
        case 2 :
            fifo();
            break;
        case 3 :
            optimal();
            break;
        case 4 :
            lru();
            break;
        case 5 :
            lfu();
            break;
        case 6 :
            secondchance();
            break;
        default :
            return 0;
            break;
    }
  }
```

}

Output

```
205120081@ca: ~
205120081@ca:~$ vi ques35.c
205120081@ca:~$ gcc ques35.c
205120081@ca:~$ ./a.out

Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:1

Enter length of page reference sequence:5

Enter the page reference sequence:
4
5
6
9
7

Enter no of frames:3

Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:2

For 4 : 4
For 5 : 4 5
For 6 : 4 5 6
For 9 : 5 6 9
For 7 : 6 9 7
Total no of page faults:5
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:3

For 4 : 4
For 5 : 5
For 6 : 6
For 9 : 9
For 7 : 7
Total no of page faults:5
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:
```

```
Enter your choice:4

For 4 : 4
For 5 : 4 5
For 6 : 4 5 6
For 9 : 9 5 6
For 7 : 9 7 6
Total no of page faults:5
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
```


```
Total no of page faults:5
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
```


```
Enter your choice:
```