A)Study of linux commands

cd command: Change to new directory.

```
(base) matlab@sjt216-0084:~$ cd desktop
(base) matlab@sjt216-0084:~/desktop$
```

mkdir command: create new directory.

```
(base) matlab@sjt216-0084:~/desktop$ mkdir sag
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag
(base) matlab@sjt216-0084:~/desktop$
```

rmdir command: remove empty directory (remove files first).

```
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag
(base) matlab@sjt216-0084:~/desktop$ rmdir sag
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt
(base) matlab@sjt216-0084:~/desktop$
```

mv command: change name of directory

```
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag  sagar
(base) matlab@sjt216-0084:~/desktop$ mv sagar sai
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag  sai
(base) matlab@sjt216-0084:~/desktop$
```

pwd command: show current directory

```
(base) matlab@sjt216-0084:~/desktop$ pwd
/home/matlab/desktop
(base) matlab@sjt216-0084:~/desktop$
```

date command: show date and time history command:
list of previously executed commands

```
(base) matlab@sjt216-0084:~/desktop$ date
Thursday 22 August 2024 04:35:52 PM IST
(base) matlab@sjt216-0084:~/desktop$
```

```
(base) matlab@sjt216-0084:~/desktop$ history
 1015  npm i jsonwebtoken
 1016  npm i nodemon
 1017  npm nodemon server.js
 1018  nodemon server.js
 1019  npm run dev
 1020  npm i node
 1021  node server.js
 1022  mongo
 1023  cd "/tmp/" && gcc tempCodeRunnerFile.c -o tempCodeRunnerFile && "/tmp/"tempCodeRunnerFile
 1024  cd "/home/matlab/" && g++ class Player:.cpp -o class Player: && "/home/matlab/"class Player:
 1025  cd "/tmp/" && gcc tempCodeRunnerFile.c -o tempCodeRunnerFile && "/tmp/"tempCodeRunnerFile
 1026  cd "/home/matlab/" && g++ class Player:.cpp -o class Player: && "/home/matlab/"class Player:
 1027  ls
 1028  gcc p1.c
 1029  ./a.out
 1030  ls
```

cal dec 2004 command: Prints a calendar for the specified month of the specified year.

```
(base) matlab@sjt216-0084:~/desktop$ cal dec 2004
    December 2004
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
```

```
(base) matlab@sjt216-0084:~/desktop$ cal 2004
                                2004
      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   1  2  3  4  5  6  7      1  2  3  4  5  6
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   7  8  9 10 11 12 13
11 12 13 14 15 16 17  15 16 17 18 19 20 21  14 15 16 17 18 19 20
18 19 20 21 22 23 24  22 23 24 25 26 27 28  21 22 23 24 25 26 27
25 26 27 28 29 30 31  29                    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  2  3                  1            1  2  3  4  5
 4  5  6  7  8  9 10   2  3  4  5  6  7  8   6  7  8  9 10 11 12
11 12 13 14 15 16 17   9 10 11 12 13 14 15  13 14 15 16 17 18 19
18 19 20 21 22 23 24  16 17 18 19 20 21 22  20 21 22 23 24 25 26
25 26 27 28 29 30     23 24 25 26 27 28 29  27 28 29 30
                      30 31

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


      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6            1  2  3  4
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   5  6  7  8  9 10 11
10 11 12 13 14 15 16  14 15 16 17 18 19 20  12 13 14 15 16 17 18
17 18 19 20 21 22 23  21 22 23 24 25 26 27  19 20 21 22 23 24 25
24 25 26 27 28 29 30  28 29 30              26 27 28 29 30 31
31
(base) matlab@sjt216-0084:~/desktop$
```

man command: show online documentation by program name

```
(base) matlab@sjt216-0084:~/desktop$ man ls
(base) matlab@sjt216-0084:~/desktop$
```

who command: who is on the system and what they are doing

```
(base) matlab@sjt216-0084:~/desktop$ who
matlab   :0           2024-08-22 11:36 (:0)
(base) matlab@sjt216-0084:~/desktop$
```

who am I command: who is logged onto this terminal

```
(base) matlab@sjt216-0084:~/desktop$ whoami
matlab
(base) matlab@sjt216-0084:~/desktop$
```

uptime command: show one line summary of system status

```
(base) matlab@sjt216-0084:~/desktop$ uptime
 16:41:35 up  5:07,  1 user,  load average: 0.15, 0.28, 0.26
(base) matlab@sjt216-0084:~/desktop$
```

tty command: know the terminal name.

```
(base) matlab@sjt216-0084:~/desktop$ tty
/dev/pts/0
(base) matlab@sjt216-0084:~/desktop$
```

uname command: print system information

```
(base) matlab@sjt216-0084:~/desktop$ uname
Linux
(base) matlab@sjt216-0084:~/desktop$
```

cat command: view files

```
(base) matlab@sjt216-0084:~/desktop$ cat > sagar.txt
sagarteja
(base) matlab@sjt216-0084:~/desktop$ cat sagar.txt
sagarteja
(base) matlab@sjt216-0084:~/desktop$
```

cp command: copy files

```
(base) matlab@sjt216-0084:~/desktop$ cat sagar.txt
sagarteja
(base) matlab@sjt216-0084:~/desktop$ cp sagar.txt saga1.txt
(base) matlab@sjt216-0084:~/desktop$ cat saga1.txt
sagarteja
(base) matlab@sjt216-0084:~/desktop$
```

ls command: list files in a directory and their attributes

```
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag  saga1.txt  sagar.txt  sai
(base) matlab@sjt216-0084:~/desktop$
```

rm command: remove files

```
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag  saga1.txt  sagar.txt  sal
(base) matlab@sjt216-0084:~/desktop$ rm saga1.txt
(base) matlab@sjt216-0084:~/desktop$ ls
demo1.txt  demo.txt  directory  documents  os1.txt  os2.txt  os.a  os.txt  rollno.txt  sag  sagar.txt  sal
(base) matlab@sjt216-0084:~/desktop$
```

head command: show first few lines of a file(s)

```
(base) matlab@sjt216-0084:~/desktop$ head sagar.txt
sagarteja
(base) matlab@sjt216-0084:~/desktop$
```

tail command: show last few lines of a file; or reverse line order

```
(base) matlab@sjt216-0084:~/desktop$ tail sagar.txt
sagarteja
(base) matlab@sjt216-0084:~/desktop$
```

b) Shell Programming

- Handling the command line arguments
- String reversal, multiplication table
- If-Else, Nested If Else, Switch cases in shell
- **Exercises:**
  1. Read three numbers from the keyboard and print the minimum value.
  2. Read in three numbers from the keyboard and print the maximum value.
  3. Swap two numbers without using third variable.
  4. Read the marks and print the grade of the student (use elif).
  5. Read two data and perform basic arithmetic operations based on User choice (use case).

QUESTION

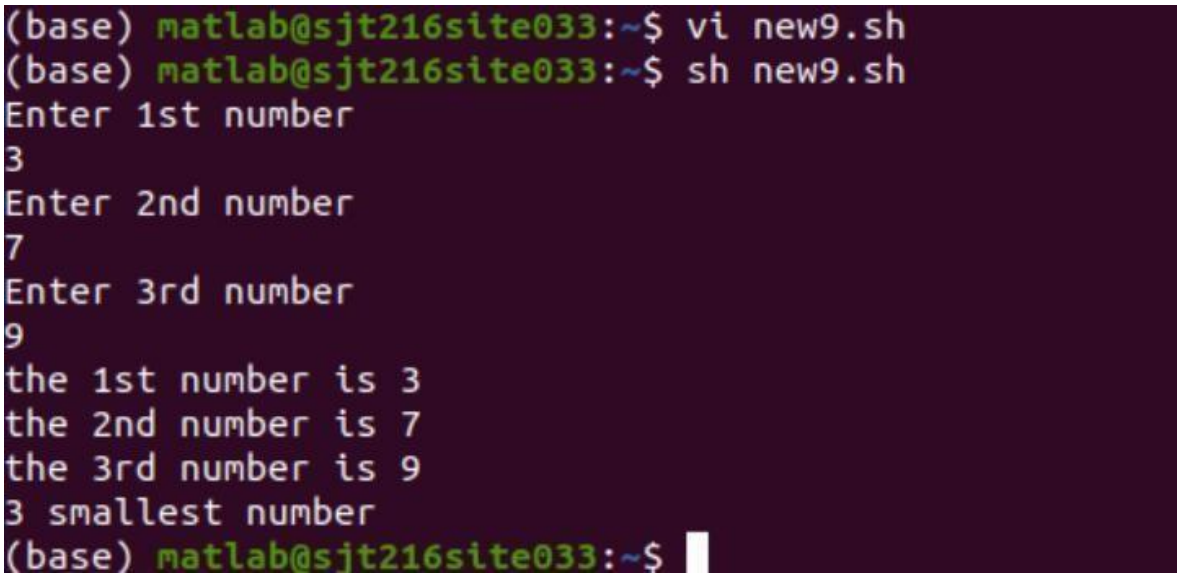
NO:1

Read three numbers from the keyboard and print the minimum value.

CODE:

```
echo "Enter three numbers" read

a read b read c if test $a -lt $b

then if test $c -lt $a then echo "$c

is minimum"

    else echo "$a is minimum"

    fi

else if test $c -lt $b then echo

   "$c is minimum" else echo

  "$b is minimum"

 fi fi
```

OUTPUT:



NO:02

Read in three numbers from the keyboard and print the maximum value.

CODE:

```
echo Enter the three number read

a read b read c if test $a -gt $b then

if test $a -gt $c then echo "$a is

greater" else echo "$c is greater"
```

QUESTION

fi

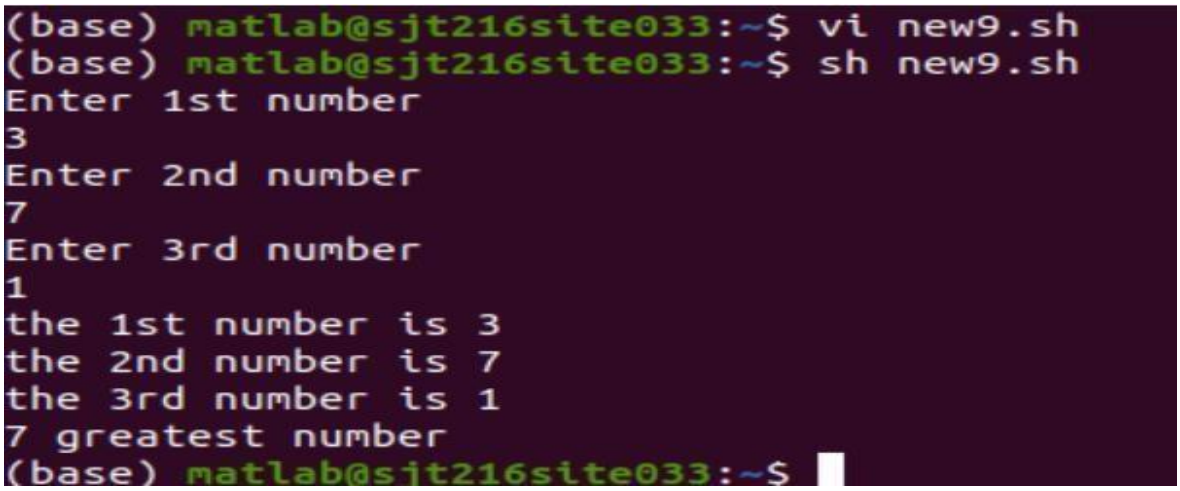
else if test $b -gt $c then echo

"$b is greater" else echo

"$c is greater"

fi fi

OUTPUT :



NO: 03

Swap two numbers without using third variable.

CODE:

echo Enter two numbers read a read b echo "

Before Swapping,Numbers are $a and $b" a=`expr $a

+ $b` b=`expr $a - $b` a=`expr $a - $b` echo " After

Swapping,Numbers are $a

and $b" OUTPUT:

QUESTION

```
(base) matlab@sjt216site034:~$ vi fileB.sh
(base) matlab@sjt216site034:~$ sh fileB.sh
enter two numbers
4
6
before swapping,numbers are 4 and 6
after swapping,numbers are 6 and 4
(base) matlab@sjt216site034:~$
```

QUESTION NO:04

Read the marks and print the grade of the student (use elif).

CODE:

```
echo "Enter the mark of the Student: "

read m if test $m -gt 100 then echo

"Enter Invalid Marks" elif test $m -gt 89

then
```

```
echo " Grade is S"
elif test $m -gt 79
then echo " Grade
is A" elif test $m gt
69 then echo "
Grade is B" elif test
$m -gt 59 then
echo " Grade is C"
elif test $m -gt 49
then echo " Grade
is D" else echo "
Grade is F"
fi
```

OUTPUT:

```
(base) matlab@sjt216site034:~$ vi fileD.sh
(base) matlab@sjt216site034:~$ sh fileD.sh
Enter the marks of the student:
58
fileD.sh: 20: Syntax error: "else" unexpected
(base) matlab@sjt216site034:~$ vi fileD.sh
(base) matlab@sjt216site034:~$ sh filrD.sh
sh: 0: Can't open filrD.sh
(base) matlab@sjt216site034:~$ sh fileD.sh
Enter the marks of the student:
58
the grade is D
Enter the marks of the student:
69
the grade is C
(base) matlab@sjt216site034:~$
```

QUESTION NO:05

Read two data and perform basic arithmetic operations based on User choice (use case).

CODE:

```
echo "Enter Two Numbers" read

a

read b echo "Select the

Operation" echo " 1 for +
```

```
echo  "  2  for  -"
echo  "  3  for  *"
echo  "  4  for  /"
echo  "  5  for  %"
read op case $op
in
          1)sum=`expr $a + $b` echo
            "$sum";;
          2)sub=`expr $a - $b` echo
            "$sub";;
          3)pro=`expr $a '*' $b` echo
            "$pro";;
          4)div=`expr $a '/' $b` echo
            "$div";;
         5)mod=`expr $a % $b` echo
   "$mod";; esac
```

OUTPUT:

```
((base) matlab@sjt216site034:~$ vi fileF.sh
((base) matlab@sjt216site034:~$ sh fileF.sh
Eenter two numbers
55
f8
(enter the operation
(1 for +
s2 for -
(3 for *
E4 for /
55 for %
t2
E-3
6(base) matlab@sjt216site034:~$ sh fileF.sh
tenter two numbers
(4
 6
 enter the operation
 1 for +
 2 for -
 3 for *
 4 for /
 5 for %
 3
 24
 (base) matlab@sjt216site034:~$
```

QUESTION NO: 01

Checking the Process Identifier

CODE:

```
#include <stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/wait.h>       int

main(void) { pid_t pid;

pid=fork(); if (pid==-1) {

perror("Error"); exit(EXIT_FAILURE);
```

```
    }
    else if(pid==0) { printf("Child Message My Id is
    %d\n",getpid()); printf("Child Message My Parent Id is
    %d\n",grtppid());
    }
    else { printf("Parent Message My Id is
   %d\n",getpid()); printf("Parent Message My Child
   Id is %d\n",pid);
    }
    return 0;
}
```

OUTPUT:

```
(base) matlab@sjt216-0084:~$ gcc p11.c
(base) matlab@sjt216-0084:~$ ./a.out
Parent Message My Id is 26683
Parent Message My Child Id is 26684
Child Message My Id is 26684
Child Message My Parent Id is 26683
(base) matlab@sjt216-0084:~$
```

QUESTION NO: 02

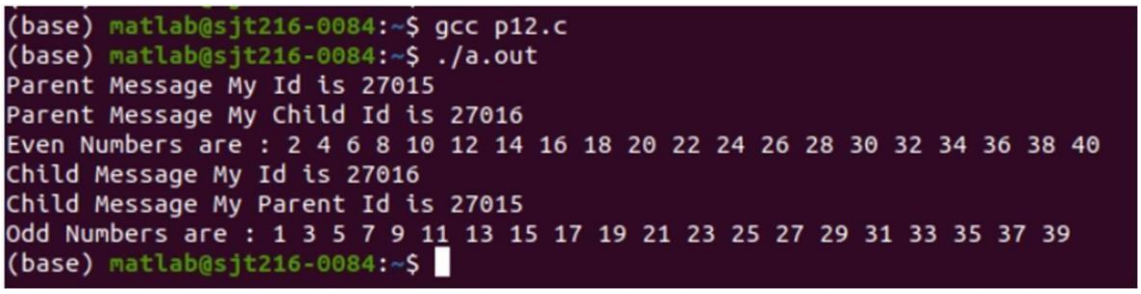
Assigning new task to child

CODE:

```
#include <stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/wait.h> int
main(void) { int n=40; int
status; pid_t pid; pid=fork();
if (pid==-1) {
perror("Error");
exit(EXIT_FAILURE);
    }
    else if(pid==0) { printf("Child Message My Id is
```

```
    %d\n",getpid()); printf("Child Message My Parent Id is
    %d\n",grtppid()); printf("Odd Numbers are : "); for(int
        i=1;i<n+1;i=i+2)
            { printf("%d ",i);
    }
    printf("\n");
    }
    else { pid_t
    i;
    printf("Parent Message My Id is %d\n",getpid()); printf("Parent
    Message My Child Id is %d\n",pid);
    printf("Even Numbers are : "); for(int
    i=2;i<n+1;i=i+2) { printf("%d
            ",i);
    }
    printf("\n");
    }
    return 0;
}
```

OUTPUT:



QUESTION NO:03

Providing the path name and program name to exec()

CODE:

Parent code:

```
#include <stdio.h>

#include<unistd.h>
```

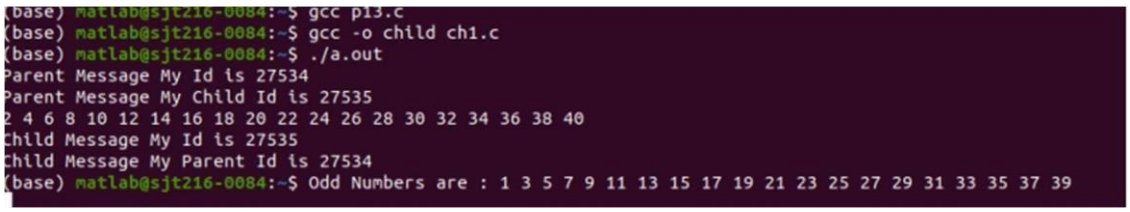
```
#include<stdlib.h>
#include<sys/wait.h> int
main(void) { int n=40; int
status; pid_t pid; pid=fork();
if (pid==-1) {
perror("Error");
exit(EXIT_FAILURE);
    }
    else if(pid==0)
    { execlp("./child","./child",NULL);
    printf("Child Message My Id is
        %d\n",getpid()); printf("Child Message My Parent Id is
    %d\n",grtppid()); printf("\n");
    }
    else { pid_t
    i;
   printf("Parent Message My Id is %d\n",getpid());
   printf("Parent Message My Child Id is %d\n",pid);
   printf("Even Numbers are : "); for(int
   i=2;i<n+1;i=i+2)
        { printf("%d ",i);
    }
    printf("\n");
    }
    return 0;
}
```

Child code:

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

```
main() { int n=40; printf("Odd

Numbers are : "); for (int

i=1;i<n+1;i=i+2) { printf("%d

",i);

}

return 0;

}
```

OUTPUT:



QUESTION NO:04

Synchronizing Parent and child process using wait()

CODE:

Parent code:

```
#include <stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/wait.h> int

main(void) { int n=40; int

status; pid_t pid; pid=fork();

if (pid==-1) {

perror("Error");

exit(EXIT_FAILURE);

    }

    else            if(pid==0)          {

        execlp("./child","./child",NULL); printf("Child  Message

        My      Id      is      %d\n",getpid()); printf("Child

    Message My Parent Id is %d\n",grtppid()); printf("\n");

    }

    else { pid_t
```

```
    i;
    i=wait(&status); printf("Parent Message My Id is
   %d\n",getpid()); printf("Parent Message My Child Id is
   %d\n",pid); for(int i=2;i<n+1;i=i+2) { printf("%d
   ",i);
    }
    printf("\n");
    }
    return 0;
}
```

Child code:

```
#include <stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/wait.h>
int main() { int n=40; for
(int i=1;i<n+1;i=i+2)
     { printf("%d ",i);
}
return 0;
}
```

OUTPUT:

```
matlab@sjt216site064:~$ vi ques2.c
matlab@sjt216site064:~$ gcc ques2.c
matlab@sjt216site064:~$ ./a.out
Child Message My ID is 13313
Child Message My PARENT ID is 13312
1  3  5  7  9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
PARENT Message My ID is 13312
PARENT Message My CHILD ID is 13313
2  4  6  8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40
```

d) CPU Scheduling

i. Implement the various process scheduling algorithms such as FCFS, SJF, Priority (Non Preemptive). **(Easy)**

FOR FCFS:

CODE:

```
#include <stdio.h> void findWaitingTime(int processes[], int n, int bt[], int
wt[], int at[])
   { int service_time[n]; service_time[0] = at[0];
   wt[0] = 0; for (int i = 1; i < n; i++) {
   service_time[i] = service_time[i-1] + bt[i-1];
   wt[i] = service_time[i] - at[i]; if (wt[i] < 0)
         { wt[i] = 0;
      }
   }
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
   { for (int i = 0; i < n; i++) { tat[i]
      = bt[i] + wt[i];
   }
}
void findAverageTime(int processes[], int n, int bt[], int at[]) { int wt[n], tat[n], total_wt
   = 0, total_tat = 0; findWaitingTime(processes, n, bt, wt, at);
   findTurnAroundTime(processes, n, bt, wt, tat); printf("Processes Arrival Time Burst
   Time Waiting Time Turnaround Time\n"); for (int i = 0; i < n; i++) { total_wt += wt[i]; total_tat
      += tat[i];
      printf("%d %d %d %d %d\n", processes[i], at[i], bt[i], wt[i], tat[i]);
   }
   printf("\nAverage waiting time = %.2f\n", (float)total_wt / n); printf("Average turnaround
   time = %.2f\n", (float)total_tat / n);
}
int main()
   { int n =
   4; int processes[] = {1, 2, 3, 4}; int arrival_time[] = {0, 2, 4,
   5}; int burst_time[] = {7, 4, 1, 4};
```

```
    findAverageTime(processes, n, burst_time, arrival_time); return

    0;

}
```

OUTPUT:

```
(base) matlab@sjt216site034:~$ vi fileH.c
(base) matlab@sjt216site034:~$ gcc fileH.c
(base) matlab@sjt216site034:~$ ./a.out
Processes Arrival Time Burst Time Waiting Time Turnaround Time
1 0 7 0 7
2 2 4 5 9
3 4 1 7 8
4 5 4 7 11

Average waiting time = 4.75
Average turnaround time = 8.75
(base) matlab@sjt216site034:~$
```

FOR SJF NON PREEMPTIVE:

CODE:

```
#include <stdio.h> void findWaitingTime(int processes[], int n, int bt[], int

wt[], int at[])

    { int completed = 0, time = 0, min_bt, shortest; int

    is_completed[n];  for  (int  i  =  0;  i  <  n;  i++)  {

    is_completed[i] = 0;

    }
    while (completed != n) {

        min_bt = 1e9; shortest = -1; for (int i = 0; i < n; i++) { if (at[i]

        <= time && !is_completed[i] && bt[i] < min_bt)

                { min_bt = bt[i]; shortest

                = i;

            }

        }

        if (shortest == -1)

            {          time++;

            continue;
```

```
        }
        wt[shortest] = time - at[shortest];
        time += bt[shortest];
        is_completed[shortest] = 1;
        completed++;
    }
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
    { for (int i = 0; i < n; i++) { tat[i]
        = bt[i] + wt[i];
    }
}
void findAverageTime(int processes[], int n, int bt[], int at[]) { int wt[n], tat[n], total_wt
    = 0, total_tat = 0; findWaitingTime(processes, n, bt, wt, at);
    findTurnAroundTime(processes, n, bt, wt, tat); printf("Processes Arrival Time Burst
    Time Waiting Time Turnaround Time\n"); for (int i = 0; i < n; i++) { total_wt += wt[i];
    total_tat += tat[i];
        printf("%d %d %d %d %d\n", processes[i], at[i], bt[i], wt[i], tat[i]);
    }
    printf("\nAverage waiting time = %.2f\n", (float)total_wt / n); printf("Average turnaround
    time = %.2f\n", (float)total_tat / n);
}
int main()
    { int n =
    4; int processes[] = {1, 2, 3, 4}; int arrival_time[] = {0, 1, 2,
    3}; int burst_time[] = {6, 8, 7, 3};
    findAverageTime(processes, n, burst_time, arrival_time); return
    0;
}
```

OUTPUT:

```
(base) matlab@sjt216site034:~$ vi fileI.c
(base) matlab@sjt216site034:~$ gcc fileI.c
(base) matlab@sjt216site034:~$ ./a.out
Processes Arrival Time Burst Time Waiting Time Turnaround Time
1 0 6 0 6
2 1 8 15 23
3 2 7 7 14
4 3 3 3 6

Average waiting time = 6.25
Average turnaround time = 12.25
(base) matlab@sjt216site034:~$
```

FOR PRIORITY NON PREEMPTIVE:

CODE:

```
#include<stdio.h

> struct Process { int

id; int arrival;

int      burst;    int

priority;          int

waiting;          int

   turnaround;

};

void findWaitingTime(struct Process proc[], int n)

   { int completed = 0, time = 0; int min_priority = 1e9; int shortest = -1; int check = 0; while

   (completed != n) { min_priority = 1e9; shortest = -1; for (int j = 0; j < n; j++) { if ((proc[j].arrival

   <= time) && (proc[j].priority < min_priority) && (proc[j].burst > 0))

            { min_priority = proc[j].priority; shortest = j;

            check = 1;

         }

      }

      if (check == 0)

         {    time++;

         continue;

      }
```

```
        time += proc[shortest].burst; proc[shortest].waiting = time -
        proc[shortest].arrival - proc[shortest].burst; proc[shortest].turnaround =
        time - proc[shortest].arrival; proc[shortest].burst = 0; completed++; check =
        0;


    }
}
void findAverageTime(struct Process proc[], int n) {
    int total_wt = 0, total_tat = 0; findWaitingTime(proc, n); printf("Processes Arrival Time
    Burst Time Priority Waiting Time Turnaround Time\n"); for (int i = 0; i < n; i++) { total_wt
    += proc[i].waiting; total_tat += proc[i].turnaround;
          printf(" %d %d %d %d %d %d\n", proc[i].id, proc[i].arrival, proc[i].burst, proc[i].priority,
proc[i].waiting, proc[i].turnaround);
    }
    printf("\nAverage waiting time = %.2f\n", (float)total_wt / n); printf("Average turnaround
    time = %.2f\n", (float)total_tat / n);
}
int main()
    { int n =
    4; struct Process proc[] = { {1, 0, 8,
    2},
                        {2, 1, 4, 1},
                        {3, 2, 9, 3},
                        {4, 3, 5, 2}
                       };
    findAverageTime(proc, n); return
    0;
}
```

OUTPUT:

```
(base) matlab@sjt216-0084:~$ gcc pnp.c
(base) matlab@sjt216-0084:~$ ./a.out
Processes Arrival Time Burst Time Priority Waiting Time Turnaround Time
    1           0           0          2          0              8
    2           1           0          1          7              11
    3           2           0          3          15              24
    4           3           0          2          9              14

Average waiting time = 7.75
Average turnaround time = 14.25
(base) matlab@sjt216-0084:~$
```

ii. Implement the various process scheduling algorithms such as SJF, Priority, Round Robin (preemptive). **(Medium)**

FOR SJF PREEMPTIVE

CODE :

```
#include <stdio.h> #include <limits.h> void findWaitingTime(int
processes[], int n, int bt[], int wt[], int at[])
  { int remaining_bt[n]; for (int
  i = 0; i < n; i++)
    { remaining_bt[i] = bt[i];
  }
  int completed = 0, time = 0, min_bt = INT_MAX;
  int shortest = 0; int finish_time; int check = 0;
  while (completed != n)
    { for (int j = 0; j < n; j++) { if ((at[j] <= time) && (remaining_bt[j] < min_bt) &&
      (remaining_bt[j] > 0)) { min_bt
        = remaining_bt[j]; shortest =
        j; check = 1;
      }
    }
    if (check == 0)
      {   time++;
      continue;
    }
    remaining_bt[shortest]--; min_bt
    =   remaining_bt[shortest];   if
```

```
    (min_bt == 0) { min_bt =
    INT_MAX;
    }
    if (remaining_bt[shortest] == 0) { completed++; check = 0;
      finish_time = time + 1; wt[shortest] = finish_time -
      bt[shortest] - at[shortest]; if (wt[shortest] < 0) {
      wt[shortest] = 0;
      }
    }
    time++;
  }
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
  { for (int i = 0; i < n; i++) { tat[i]
    = bt[i] + wt[i];
  }
}
void findAverageTime(int processes[], int n, int bt[], int at[]) { int wt[n], tat[n], total_wt
  = 0, total_tat = 0; findWaitingTime(processes, n, bt, wt, at);
  findTurnAroundTime(processes, n, bt, wt, tat); printf("Processes Arrival Time Burst
  Time Waiting Time Turnaround Time\n"); for (int i = 0; i < n; i++) { total_wt += wt[i];
  total_tat += tat[i];
    printf("%d %d %d %d %d\n", processes[i], at[i], bt[i], wt[i], tat[i]);
  }
  printf("\nAverage waiting time = %.2f\n", (float)total_wt / n); printf("Average turnaround
  time = %.2f\n", (float)total_tat / n);
}
int main()
  { int n =
  4; int processes[] = {1, 2, 3, 4}; int arrival_time[] = {0, 1, 2,
  3}; int burst_time[] = {6, 8, 7, 3};
```

```
    findAverageTime(processes, n, burst_time, arrival_time); return

    0;

}
```

OUTPUT:

FOR PROIRITYPREEMPTIVE

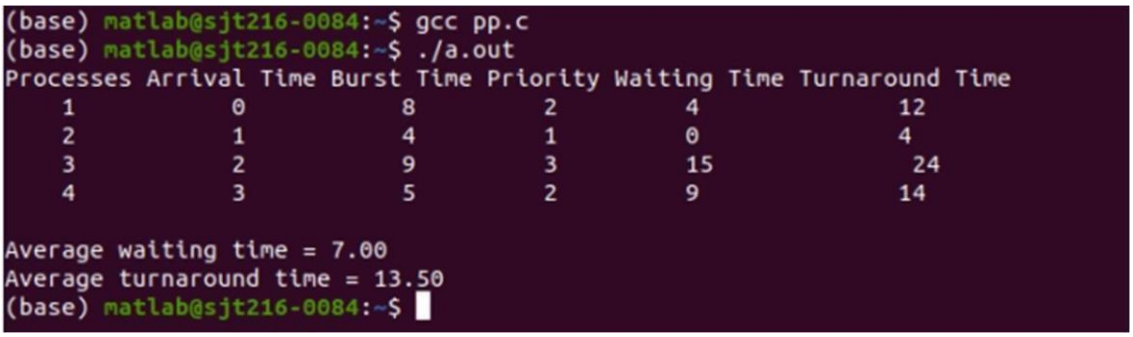
CODE:

```
#include   <stdio.h>

#include  <limits.h>

struct Process

{ int id; int arrival;

   int    burst;     int

   priority;        int

   waiting;         int

   turnaround; int remaining;

};

void findWaitingTime(struct Process proc[], int n)
{

   int   time      =          0;

         int completed = 0; int

   min_priority;              int

   shortest = -1; int check

   = 0; while (completed

   != n)

   {
```

```
        min_priority = INT_MAX;
        shortest = -1; for (int j = 0;
        j < n; j++)
        {
            if (proc[j].arrival <= time && proc[j].remaining > 0 && proc[j].priority < min_priority)
            {
                min_priority = proc[j].priority; shortest
                = j; check = 1;
            }
        }
        if (check == 0)
        {
            time++; continue;
        } proc[shortest].remaining--; if
        (proc[shortest].remaining == 0)
        {
            completed++; check = 0; int finish_time = time + 1; proc[shortest].waiting =
            finish_time - proc[shortest].burst - proc[shortest].arrival; proc[shortest].turnaround =
            finish_time - proc[shortest].arrival; if
            (proc[shortest].waiting < 0)
            {
                proc[shortest].waiting = 0;
            }
        }
        time++;
    }
}
void findAverageTime(struct Process proc[], int n)
{
    int total_wt = 0, total_tat = 0; findWaitingTime(proc, n); printf("Processes Arrival Time Burst
    Time Priority Waiting Time Turnaround Time\n"); for (int i = 0; i < n; i++)
```

```
    {
        total_wt += proc[i].waiting; total_tat += proc[i].turnaround; printf("
        %d %d %d %d %d %d\n", proc[i].id, proc[i].arrival, proc[i].burst,
        proc[i].priority, proc[i].waiting, proc[i].turnaround);
    }
    printf("\nAverage waiting time = %.2f\n", (float)total_wt / n); printf("Average turnaround
    time = %.2f\n", (float)total_tat / n);
}
int main()
{
    int n = 4; struct Process proc[]
    = {
        {1, 0, 8, 2},
        {2, 1, 4, 1},
        {3, 2, 9, 3},
        {4, 3, 5, 2}}; for (int i
    = 0; i < n; i++)
    {
        proc[i].remaining = proc[i].burst;
    }
    findAverageTime(proc, n); return
    0;
}
```

OUTPUT:



FOR ROUND ROBIN:

CODE:

```
#include <stdio.h> void findWaitingTime(int processes[], int n, int bt[], int wt[], int at[],

int quantum)

{

   int remaining_bt[n]; for

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

   {

      remaining_bt[i] = bt[i];

   }

   int time = 0; while

   (1)

   {

      int done = 1; for (int i =

      0; i < n; i++)

      {

         if (remaining_bt[i] > 0)

         {

            done = 0; if (remaining_bt[i] > quantum && at[i]

            <= time)

            {

               time += quantum; remaining_bt[i] -

               = quantum;

            }

            else if (at[i] <= time)

            {

               time  +=  remaining_bt[i];

               wt[i] = time - bt[i] - at[i];

               remaining_bt[i] = 0;

            }

            else

            {
```

```
                time++;
            }
        }
    }
    if (done == 1) break;
  }
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{ for (int i = 0; i < n; i++)
  { tat[i] = bt[i] + wt[i];
  }
}
void findAverageTime(int processes[], int n, int bt[], int at[], int quantum)
{
  int wt[n], tat[n], total_wt = 0, total_tat = 0; findWaitingTime(processes, n, bt, wt,
  at, quantum); findTurnAroundTime(processes, n, bt, wt, tat); printf("Processes
  Arrival Time Burst Time Waiting Time Turnaround Time\n"); for (int i = 0; i < n;
  i++)
  {
    total_wt += wt[i]; total_tat += tat[i]; printf(" %d %d %d %d %d\n", processes[i],
    at[i], bt[i], wt[i], tat[i]);
  }
  printf("\nAverage waiting time = %.2f\n", (float)total_wt / n); printf("Average turnaround
  time = %.2f\n", (float)total_tat / n);
}
int main()
{
  int n = 4; int processes[] = {1, 2,
  3, 4}; int arrival_time[] = {0, 1,
```

```
2, 3}; int burst_time[] = {8, 4, 9, 5}; int quantum = 3;
findAverageTime(processes, n, burst_time, arrival_time, quantum); return
0;
}
```

OUTPUT:

```
(base) matlab@sjt216-0084:~$ gcc rr.c
(base) matlab@sjt216-0084:~$ ./a.out
Processes Arrival Time Burst Time Waiting Time Turnaround Time
   1          0          8          15          23
   2          1          4          11          15
   3          2          9          15          24
   4          3          5          13          18

Average waiting time = 13.50
Average turnaround time = 20.00
(base) matlab@sjt216-0084:~$
```

iii. Consider a corporate hospital where we have n number of patients waiting for consultation. The amount of time required to serve a patient may vary, say 10 to 30 minutes. If a patient arrives with an emergency, he /she should be attended immediately before other patients, which may increase the waiting time of other patients. If you are given this problem with the following algorithms, how would you devise an effective scheduling so that it optimizes the overall performance such as minimizing the waiting time of all patients. [Single queue or multi-level queue can be used].

- Consider the availability of single and multiple doctors
- Assign top priority for patients with emergency case, women, children, elders, and youngsters.
- Patients coming for review may take less time than others. This can be taken into account while using SJF.

1. FCFS
2. SJF (primitive and non-pre-emptive) **(High)**

WITH FCFS:

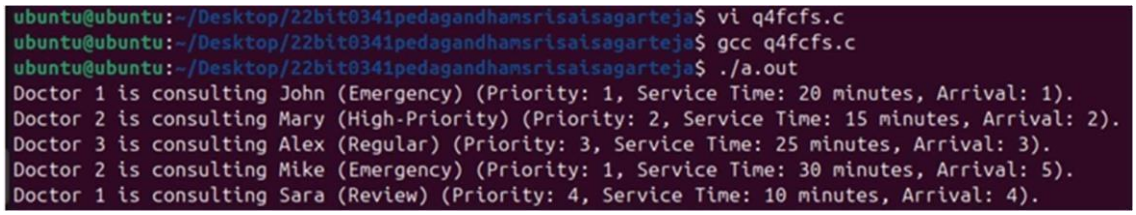
CODE:

```
#include <stdio.h> #include
<stdlib.h>
#include <string.h>
#define MAX_PATIENTS 100
#define MAX_DOCTORS 3
typedef struct {
    char name[50];
```

```
    int priority; int
    service_time; int
    arrival_time; int
    is_assigned;
  } Patient;
Patient queue[MAX_PATIENTS]; int total_patients = 0; int
  doctor_busy[MAX_DOCTORS] = {0}; int doctor_service_time[MAX_DOCTORS]
  = {0}; void add_patient(char* name, int priority, int service_time, int
  arrival_time)
    { strcpy(queue[total_patients].name, name);
    queue[total_patients].priority = priority;
    queue[total_patients].service_time = service_time;
    queue[total_patients].arrival_time = arrival_time;
    queue[total_patients].is_assigned = 0; total_patients++;
  }
  int find_next_patient(int current_time) { int selected_patient = -1; for (int i = 0; i < total_patients; i++) { if
    (queue[i].is_assigned == 0 && queue[i].arrival_time <= current_time) { if
        (selected_patient == -1 || queue[i].priority <
    queue[selected_patient].priority ||
          (queue[i].priority == queue[selected_patient].priority &&
        queue[i].arrival_time < queue[selected_patient].arrival_time))
          { selected_patient = i;
        }
      }
    }
    return selected_patient;
  }
  void simulate_doctor_schedule()
    { int current_time = 0; int patients_served = 0;
    while (patients_served < total_patients) { for (int
    doc = 0; doc <
    MAX_DOCTORS; doc++)
```

```
        { if (doctor_busy[doc] == 0) { int patient_index = find_next_patient(current_time); if
          (patient_index !=
          -1) {
               printf("Doctor %d is consulting %s (Priority: %d, Service Time: %d minutes, Arrival:
%d).\n",
                  doc + 1, queue[patient_index].name, queue[patient_index].priority,
            queue[patient_index].service_time, queue[patient_index].arrival_time);
            doctor_busy[doc] = 1; doctor_service_time[doc] =
            queue[patient_index].service_time; queue[patient_index].is_assigned = 1;
            patients_served++;
          }
        }
      }
      for (int doc = 0; doc < MAX_DOCTORS; doc++)
        { if (doctor_busy[doc] == 1) {
          doctor_service_time[doc]--; if
          (doctor_service_time[doc] == 0)
            { doctor_busy[doc] = 0;
          }
        }
      }
      current_time++;
    }
}
int main() { add_patient("John (Emergency)",
    1, 20, 1); add_patient("Mike (Emergency)",
    1, 30, 5); add_patient("Mary (HighPriority)", 2, 15, 2);
    add_patient("Alex
    (Regular)", 3, 25, 3); add_patient("Sara
    (Review)", 4, 10, 4);
    simulate_doctor_schedule(); return 0;
}
```

OUTPUT:



WITH SJF NON PREMPTIVE:

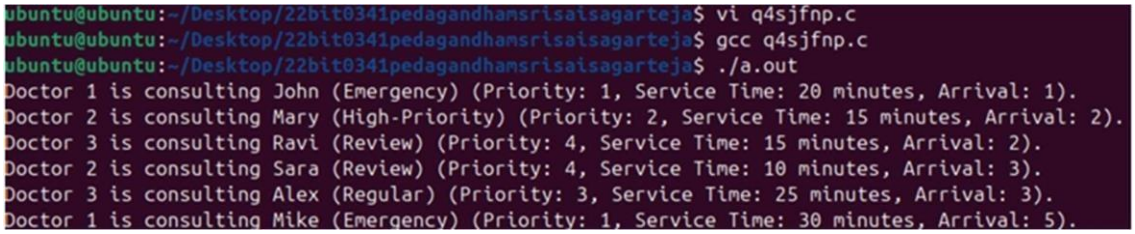
CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_PATIENTS 100

#define MAX_DOCTORS 3

typedef struct { char

name[50]; int priority; int

service_time; int

arrival_time; int is_assigned;

} Patient; Patient queue[MAX_PATIENTS]; int

total_patients = 0; int

doctor_busy[MAX_DOCTORS] = {0}; int doctor_service_time[MAX_DOCTORS] = {0};

void add_patient(char* name, int priority, int service_time, int arrival_time)

    { strcpy(queue[total_patients].name, name);

    queue[total_patients].priority = priority;

    queue[total_patients].service_time = service_time;

    queue[total_patients].arrival_time = arrival_time;

    queue[total_patients].is_assigned = 0; total_patients++;

}

int find_next_patient_sjf(int current_time) { int selected_patient = -1; for (int i = 0; i < total_patients; i++)

    { if (queue[i].is_assigned == 0 && queue[i].arrival_time <= current_time) { if (selected_patient == -1

    || queue[i].service_time < queue[selected_patient].service_time ||
```

```
            (queue[i].service_time == queue[selected_patient].service_time && queue[i].arrival_time <
 queue[selected_patient].arrival_time))
            { selected_patient = i;
          }
       }
    }
    return selected_patient;
 }
 void simulate_doctor_schedule() { int current_time
    =  0;  int  patients_served  =  0;  while
    (patients_served < total_patients) { for (int doc =
    0; doc < MAX_DOCTORS; doc++)
          { if (doctor_busy[doc] == 0) { int patient_index = find_next_patient_sjf(current_time); if
             (patient_index !=
             -1) {
                printf("Doctor %d is consulting %s (Priority: %d, Service Time: %d minutes, Arrival:
%d).\n", doc + 1, queue[patient_index].name, queue[patient_index].priority,
queue[patient_index].service_time, queue[patient_index].arrival_time); doctor_busy[doc] = 1;
doctor_service_time[doc] = queue[patient_index].service_time; queue[patient_index].is_assigned = 1;
patients_served++;
             }
          }
       }
       for (int doc = 0; doc < MAX_DOCTORS; doc++)
          {  if  (doctor_busy[doc]  ==  1)  {
             doctor_service_time[doc]--;  if
             (doctor_service_time[doc] == 0)
                { doctor_busy[doc] = 0;
             }
          }
       }
       current_time++;
```

```
    }
}
int main() { add_patient("John (Emergency)", 1,
    20, 1); add_patient("Mike (Emergency)", 1, 30,
    5); add_patient("Mary (High-Priority)", 2, 15, 2);
    add_patient("Alex (Regular)", 3, 25, 3);
    add_patient("Sara      (Review)",          4,
        10,        3); add_patient("Ravi
        (Review)",          4,          15,        2);
    simulate_doctor_schedule(); return 0;
}
```

OUTPUT:



FOR SJF PREMPTIVE:

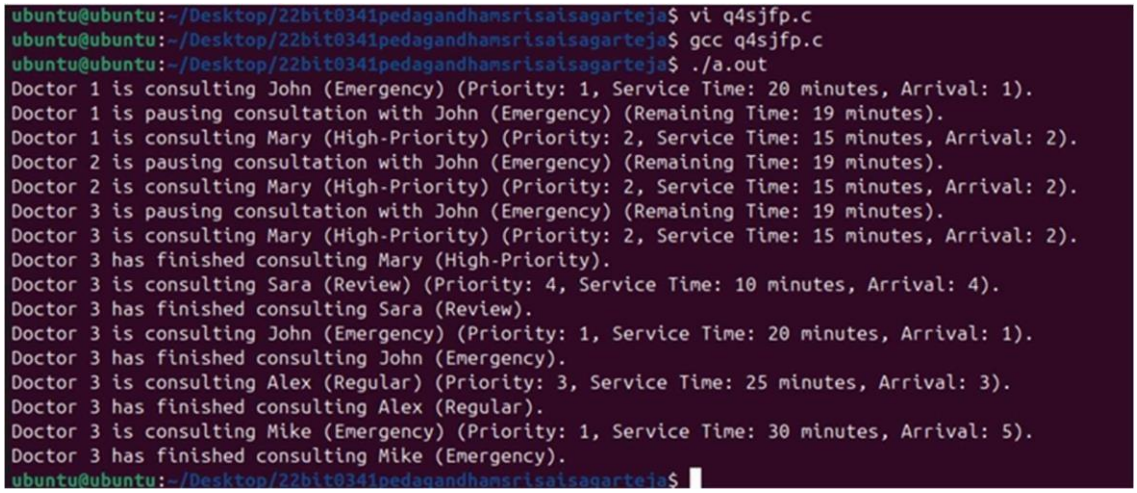
CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_PATIENTS 100
#define MAX_DOCTORS 3
typedef          struct    {
        char name[50]; int
        priority;  int
service_time;    int
remaining_time; int
arrival_time;    int
is_assigned;
} Patient;
Patient      queue[MAX_PATIENTS];      int      total_patients      =      0;      int
```

```
doctor_busy[MAX_DOCTORS] = {0}; int doctor_patient_index[MAX_DOCTORS]
= {-1}; void add_patient(char* name, int priority, int service_time, int
arrival_time)
   { strcpy(queue[total_patients].name, name);
   queue[total_patients].priority = priority;
   queue[total_patients].service_time = service_time;
   queue[total_patients].remaining_time = service_time;
   queue[total_patients].arrival_time = arrival_time;
   queue[total_patients].is_assigned = 0; total_patients++;
}
int find_next_patient_sjf_preemptive(int current_time)
   { int selected_patient = -1; for (int i = 0; i < total_patients; i++) { if
   (queue[i].arrival_time <= current_time && queue[i].remaining_time > 0)
        { if (selected_patient == -1 || queue[i].remaining_time <
queue[selected_patient].remaining_time)
           { selected_patient = i;
        }
      }
   }
   return selected_patient;
}
void simulate_doctor_schedule() { int current_time =
   0; int patients_served = 0; while (patients_served <
   total_patients) { for (int doc = 0; doc <
   MAX_DOCTORS; doc++) { int current_patient = doctor_patient_index[doc]; int
        new_patient_index = find_next_patient_sjf_preemptive(current_time);
        if (new_patient_index != -1 && (current_patient == -1 ||
queue[new_patient_index].remaining_time < queue[current_patient].remaining_time))
           { if (current_patient != -1 && queue[current_patient].remaining_time > 0) { printf("Doctor
              %d is pausing consultation with %s (Remaining Time: %d minutes).\n", doc + 1,
              queue[current_patient].name, queue[current_patient].remaining_time);
              queue[current_patient].is_assigned = 0;
```

```
            }
            doctor_busy[doc] = 1; doctor_patient_index[doc] = new_patient_index;
            queue[new_patient_index].is_assigned = 1; printf("Doctor %d is consulting %s (Priority: %d,
            Service Time: %d minutes, Arrival: %d).\n", doc + 1, queue[new_patient_index].name,
            queue[new_patient_index].priority, queue[new_patient_index].service_time,
            queue[new_patient_index].arrival_time);
          }
        }
        for (int doc = 0; doc < MAX_DOCTORS; doc++)
          { if (doctor_busy[doc] == 1) { int current_patient = doctor_patient_index[doc];
            if (current_patient != -1 && queue[current_patient].remaining_time > 0)
              {   queue[current_patient].remaining_time--;   if
              (queue[current_patient].remaining_time == 0) {
                printf("Doctor %d has finished consulting %s.\n", doc + 1,
queue[current_patient].name);
                doctor_busy[doc] = 0;
                doctor_patient_index[doc] = -1; patients_served++;
              }
            }
          }
        }
        current_time++;
    }
}
int main() { add_patient("John (Emergency)", 1,
    20, 1); add_patient("Mike (Emergency)", 1, 30,
    5); add_patient("Mary (High-Priority)", 2, 15, 2);
    add_patient("Alex (Regular)", 3, 25, 3);
    add_patient("Sara (Review)", 4,
        10, 4); simulate_doctor_schedule();
    return 0;
}
```

OUTPUT:

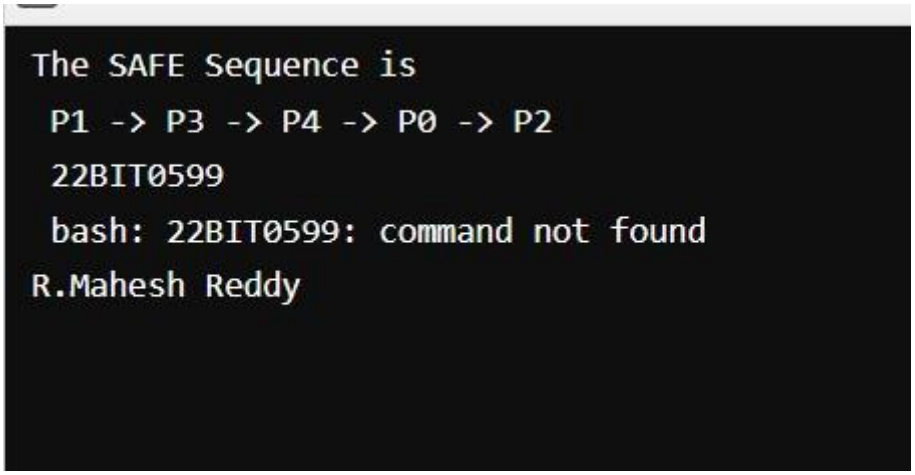

ubuntu@ubuntu:~/Desktop/22bit0341pedagandhamsrisaisagarteja$ vi q4sjfp.c
ubuntu@ubuntu:~/Desktop/22bit0341pedagandhamsrisaisagarteja$ gcc q4sjfp.c
ubuntu@ubuntu:~/Desktop/22bit0341pedagandhamsrisaisagarteja$ ./a.out
Doctor 1 is consulting John (Emergency) (Priority: 1, Service Time: 20 minutes, Arrival: 1).
Doctor 1 is pausing consultation with John (Emergency) (Remaining Time: 19 minutes).
Doctor 1 is consulting Mary (High-Priority) (Priority: 2, Service Time: 15 minutes, Arrival: 2).
Doctor 2 is pausing consultation with John (Emergency) (Remaining Time: 19 minutes).
Doctor 2 is consulting Mary (High-Priority) (Priority: 2, Service Time: 15 minutes, Arrival: 2).
Doctor 3 is pausing consultation with John (Emergency) (Remaining Time: 19 minutes).
Doctor 3 is consulting Mary (High-Priority) (Priority: 2, Service Time: 15 minutes, Arrival: 2).
Doctor 3 has finished consulting Mary (High-Priority).
Doctor 3 is consulting Sara (Review) (Priority: 4, Service Time: 10 minutes, Arrival: 4).
Doctor 3 has finished consulting Sara (Review).
Doctor 3 is consulting John (Emergency) (Priority: 1, Service Time: 20 minutes, Arrival: 1).
Doctor 3 has finished consulting John (Emergency).
Doctor 3 is consulting Alex (Regular) (Priority: 3, Service Time: 25 minutes, Arrival: 3).
Doctor 3 has finished consulting Alex (Regular).
Doctor 3 is consulting Mike (Emergency) (Priority: 1, Service Time: 30 minutes, Arrival: 5).
Doctor 3 has finished consulting Mike (Emergency).
ubuntu@ubuntu:~/Desktop/22bit0341pedagandhamsrisaisagarteja$

Code for bankers algorithm:

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

   int n, m, i, j, k;
   n = 5; m = 3;
  // Allocation matrix int
 alloc[5][3] = {{0, 1, 0},
                {2, 0, 0},
                {3, 0, 2},
                {2, 1, 1},
                {0, 0, 2}};

   // MAX Matrix int
 max[5][3] = {{7, 5, 3},
              {3, 2, 2},
              {9, 0, 2},
              {2, 2, 2},
              {4, 3, 3}};

 // Available Resources
  int avail[3] = {3, 3, 2};

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

OUTPUT:



Q1)

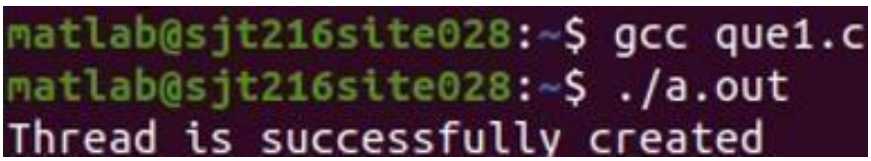
1. Write a program to create a thread and perform the following
   - Create a thread runner function
   - Set thread attributes
   - Join the parent and thread
   - Wait for thread to complete

CODE:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h> void*
q1()
```

```
{ printf("Thread is successfully
 created\n");
 return NULL;
} int
main()
{ pthread_t
 thread_id;
 int r;
 r =
 pthread_create(&thread_id,NULL,&q1,NULL);
 if(r!=0) { perror("Error");
 exit(EXIT_FAILURE);
 }
 r = pthread_join(thread_id,NULL);
 if(r!=0){
 perror("Error2");
 exit(EXIT_FAILURE);
 }
 pthread_exit(0);
}
```

OUTPUT:



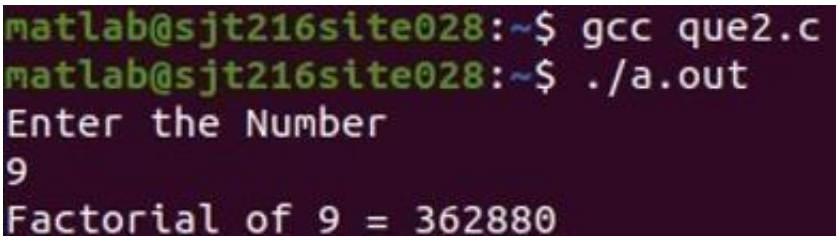
2. Write a program to create a thread to find the factorial of a natural number ‘n’.

CODE:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
void* fact(void* t){
int i,n = *(int *)t;
for(i=n;i>1;i--)
 n = n*(i-1);  printf("Factorial of %d =
 %d\n",*(int *)t,n);  return NULL;
```

```
} int
main()
{ pthread_t
 thread_id;
 int n,ret;  printf("Enter the
 Number\n");
 scanf("%d",&n);
 ret =
 pthread_create(&thread_id,NULL,&fact,&n);
 if(ret!=0) {
 perror("Error");
 exit(EXIT_FAILURE);
 }
 pthread_exit(0);
}
```

OUTPUT:



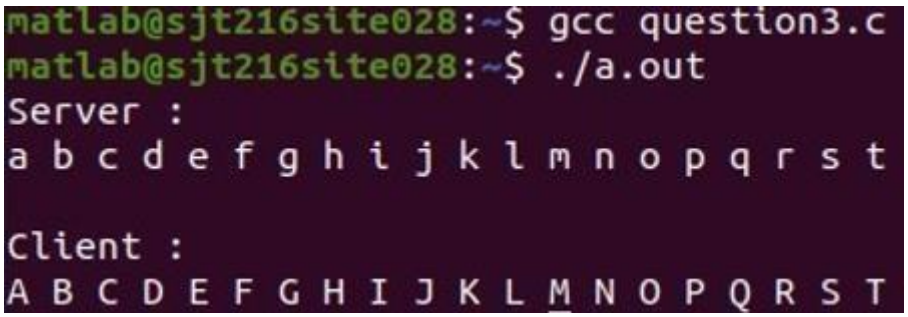
3. Assume that two processes named client and server running in the system. It is required that these two processes should communicate with each other using shared memory concept. The server writes alphabets from a..z to the shared memory .the client should read the alphabets from the shared memory and convert it to A…Z. Write a program to demonstrate the above mentioned scenario.

CODE:

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

```
#include<ctype.h>
char a[26]; void*
server(){
printf("Server : \n");
int i; char p = 'a';
for(i=0;i<26;i++)
{  a[i] =
p;  p++;
}
for(i=0;i<26;i++){
printf("%c ",a[i]);
} printf("\n");
return
NULL;
}  void*
client(){
printf("\n");
printf("Client : \n");
int i;
for(i=0;i<26;i++){
printf("%c ",toupper(a[i]));
} printf("\n");
return
NULL;
} int main(){
pthread_t thread_id;
int r,s;
r = pthread_create(&thread_id, NULL, &server, NULL);
s = pthread_create(&thread_id, NULL, &client, NULL);
if(r!=0 && s!=0){ perror("Error");
exit(EXIT_FAILURE);
}
pthread_exit(0);
}
```

OUTPUT:



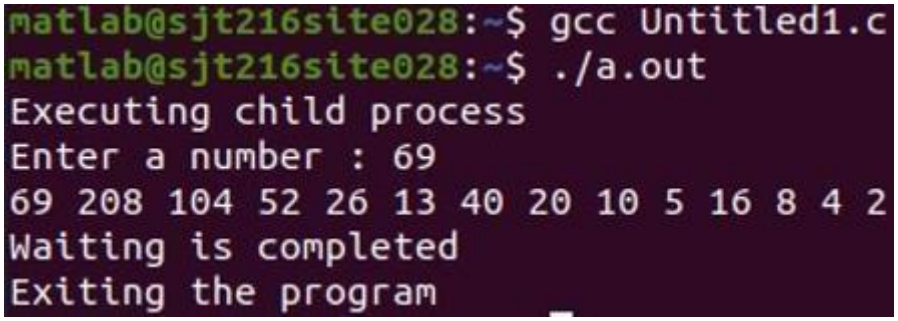
4. The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm: n = n/2, if n is even n = 3 × n + 1, if n is odd The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, if n = 35, the sequence is 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1.Write a C program using the fork () system call that generates this sequence in the child process. The starting number will be provided from the command line. For example, if 8 is passed as a parameter on the Command line, the child process will output 8, 4, 2, 1. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the wait () call to wait for the child process to complete before exiting the program.

CODE:

```
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
 #include<sys/wait.h
 > int main(void) {
 pid_t       pid;   pid=
 fork(); int status; if
 (pid==-1) {
  perror("Error");
  exit(EXIT_FAILURE);
 }  else if (pid==0) {
 printf("Executing child
 process\n"); int n;
 printf("Enter a number : ");
scanf("%d",&n)
;  while  (n>1)  {
```

```
printf("%d   ",n);
if (n%2==0) {
 n=n/2; }
 else {
 n=n*3+1;
 } }
 printf("%d\n",1);
 } else
 {
 pid_t
 i;
 i=wai
 t(0);
 printf
 ("Wa
 iting
 is
 comp
 leted
 ");
 printf
 ("\nE
 xitin
 g the
 progr
 am\n
 ");
 } return
 0;
}
```

OUTPUT:



5. Write a multithreaded program that calculates various statistical values for a list of numbers. This program will be passed a series of numbers on the command line and will then create three separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your program is passed the integers 90 81 78 95 79 72 85 , the program will report the average value as 82. The minimum value as 72. The maximum value as 95. The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the parent thread will output the values once the workers have exited.

CODE:

```
#include<pthread.h>
#include<stdlib.h>
#include<stdio.h> int
maximum,minimum;
int n;
int     average;
int    arr[100];
void* maxi()
 {
maximum=arr[0];   for
(int  i=1;i<n;i++)  {  if
(maximum<arr[i])     {
maximum=arr[i];
 } }  printf("Thread for maximum is created
successfully\n");  printf("maximum is %d
\n",maximum); return NULL;  } void* mini() {
 minimum=arr[0]; for
 (int i=1;i<n;i++) { if
 (minimum>arr[i]) {
 minimum=arr[i];
 } }  printf("Thread for minimum is created
 successfully\n");  printf("minimum is %d
 \n",minimum); return NULL;  } void* avge() {
```

```
 for (int i=0;i<n;i++) {
 average+=(arr[i]);
 }
 average=average/n;  printf("Thread for average is
 created successfully\n"); printf("average is %d
 \n",average);  return NULL;
} int main() {  pthread_t
thread_id1,thread_id2,thread_id3; int res;
printf("Enter no of Values: ");
scanf("%d",&n);        for     (int i=0;i<n;i++)
        {
 scanf("%d",&arr[i]);
 }
 res=pthread_create(&thread_id1,NULL,&maxi,NULL)
 ; if (res!=0) { perror("ERROR");
 exit(EXIT_FAILURE);
 }
 res=pthread_create(&thread_id2,NULL,&mini,NULL)
 ; if (res!=0) { perror("ERROR");
 exit(EXIT_FAILURE);
 }
 res=pthread_create(&thread_id3,NULL,&avge,NULL)
 ; if (res!=0) { perror("ERROR");
 exit(EXIT_FAILURE);
 }

 pthread_exit(0);
}
```

OUTPUT:

```
matlab@sjt216site028:~$ gcc que5.c -lpthrea
matlab@sjt216site028:~$ ./a.out
Enter no of Values: 7
90 81 78 95 79 72 85
Thread for maximum is created successfully
maximum is 95
Thread for minimum is created successfully
minimum is 72
Thread for average is created successfully
average is 82
```

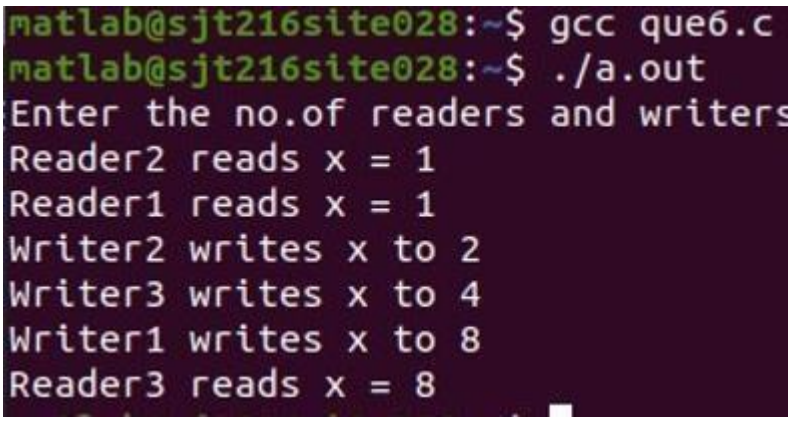
Implement the solution for reader – writer's problem.

CODE:

```
#include <stdio.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
sem_t wrt;
pthread_mutex_t mutex;
int x = 1,readcount = 0;
void reader(void *arg){
int n = rand()%5;
sleep(n);
 pthread_mutex_lock(&mutex);
 readcount++; if(readcount == 1)
 sem_wait(&wrt);
 pthread_mutex_unlock(&mutex)
 ;
 printf("Reader%d reads x = %d\n",*((int *)arg),x);
 pthread_mutex_lock(&mutex);
 readcount--;
 if(readcount==0) sem_post(&wrt);
 pthread_mutex_unlock(&mutex);
}
void writer(void *arg){
 int n = rand()%5;
 sleep(n); sem_wait(&wrt); x *= 2;
 printf("Writer%d writes x to %d\n",*((int
 *)arg),x);  sem_post(&wrt);
} void
main(){
 int nr,nw;  printf("Enter the no.of readers and
 writers: ");  scanf("%d %d",&nr,&nw);
 pthread_t r[nr],w[nw]; sem_init(&wrt,0,1);
 pthread_mutex_init(&mutex,NULL);
 int a[20],i; for( i=0;i<20;i++)
 a[i]
=                    i+1;
```

```
for(i=0;i<nw;i++
)
pthread_create(&w[i], NULL, (void *)writer, (void
*)&a[i]); for(i=0;i<nr;i++) pthread_create(&r[i], NULL,
(void *)reader, (void *)&a[i]); for(i=0;i<nw;i++)
pthread_join(w[i],NULL); for(i=0;i<nr;i++)
pthread_join(r[i],NULL); sem_destroy(&wrt);
pthread_mutex_destroy(&mutex);
}
```

OUTPUT:

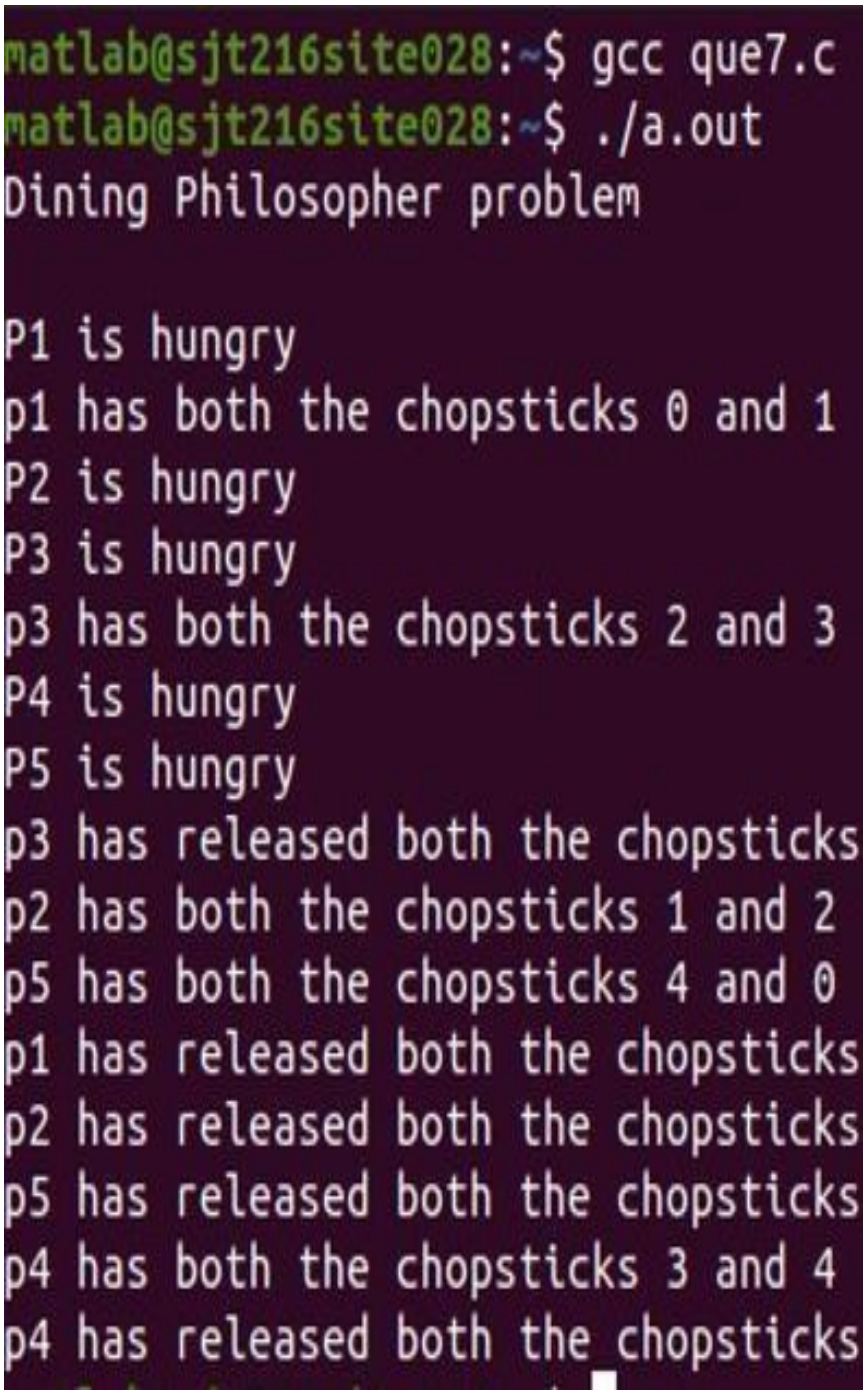


7. Implement the solution for dining philosopher's problem CODE:

```
#include <pthread.h>
#include <stdio.h>
#include <semaphore.h>
#include  <unistd.h>
#include  <stdlib.h> sem_t
c[5]; pthread_t p[5]; void
*dine(void
*args){  int i= *((int *)args);  printf("P%d is hungry\n",i+1);
sem_wait(&c[i]); sem_wait(&c[(i+1)%5]); printf("p%d has both
the chopsticks %d and %d\n",i+1,i,(i+1)%5); sleep(2);
sem_post(&c[i]); sem_post(&c[(i+1)%5]); printf("p%d has
released both the chopsticks\n",i+1);
} void
main(){
printf("Dining Philosopher problem\n\n");
int  i,a[5];       for(i=0;i<5;i++){
sem_init(&c[i],0,1);
a[i] = i; }
for(i=0;i<5;i++)
```

```
 pthread_create(&p[i],NULL,(void *)dine,(void
*)&a[i]); for(i=0;i<5;i++) pthread_join(p[i],NULL);
for(i=0;i<5;i++) sem_destroy(&c[i]); }
```

OUTPUT:



8.A pair of processes involved in exchanging a sequence of integers. The number of integers that can be produced and consumed at a time is limited to 100. Write a Program to implement the producer and consumer problem using POSIX semaphore for the above scenario.

CODE:

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

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>
#define BUFFER_SIZE 100  int
buffer[BUFFER_SIZE];
int in = 0;
int  out  =  0;  sem_t  empty;
sem_t  full;  sem_t  mutex;
void* producer(void* arg) {
int item;
 for (int i = 0; i < 200; i++) { item =
 rand() % 100; sem_wait(&empty);
 sem_wait(&mutex); buffer[in] = item;
 printf("Producer produced: %d\n",
 item); in = (in + 1) % BUFFER_SIZE;
 sem_post(&mutex);  sem_post(&full);
 sleep(1);
 }
 return NULL;
} void* consumer(void* arg)
 { int item;  for (int i = 0; i <
 200; i++) {
 sem_wait(&full);
 sem_wait(&mutex); item =
 buffer[out];
 printf("Consumer
 consumed: %d\n", item); out = (out +
 1) % BUFFER_SIZE;
 sem_post(&mutex);
 sem_post(&empty); sleep(1);
 }
 return NULL;
} int
main() {
pthread_t              producer_thread,              consumer_thread;
sem_init(&empty,  0,  BUFFER_SIZE);  sem_init(&full,  0,  0);
sem_init(&mutex,  0,  1);  pthread_create(&producer_thread,
NULL,  producer,  NULL);  pthread_create(&consumer_thread,
```

```
NULL, consumer, NULL); pthread_join(producer_thread,
NULL); pthread_join(consumer_thread, NULL);
 sem_destroy(&empty); sem_destroy(&full);
 sem_destroy(&mutex)
; return 0; }
```

OUTPUT:

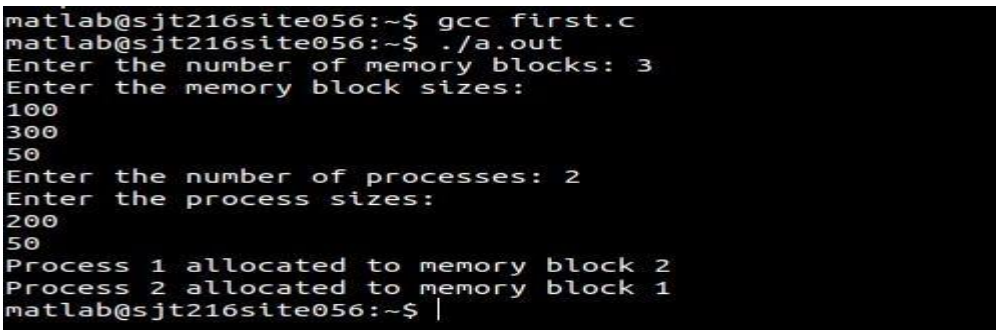
```
~$ ./a.out
Producer produced: 83
Consumer consumed: 83
Producer produced: 86
Consumer consumed: 86
Producer produced: 77
Consumer consumed: 77
Producer produced: 15
Consumer consumed: 15
Producer produced: 93
Consumer consumed: 93
Producer produced: 35
Consumer consumed: 35
Producer produced: 86
Consumer consumed: 86
Producer produced: 92
Consumer consumed: 92
^C
```

QUESTION -1

FIRST FIT CODE:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 100
int main() {
   int memory[MAX_SIZE], n, process_size[MAX_SIZE], m, i, j, first_fit_index;
printf("Enter the number of memory blocks: ");     scanf("%d",&n);
   printf("Enter the memory block sizes: \n");
for(i=0;i<n;i++) {
scanf("%d",&memory[i]);
   }
   printf("Enter the number of processes: ");
scanf("%d",&m);
   printf("Enter the process sizes: \n");
for(i=0;i<m;i++) {
      scanf("%d",&process_size[i]);
   }
   for(i=0;i<m;i++) {
first_fit_index=-1;        for(j=0;j<n;j++) {
         if(memory[j] >= process_size[i]) {
            first_fit_index=j;
            break;
         }
      }
      if(first_fit_index !=-1) {
         memory[first_fit_index] -= process_size[i];
         printf("Process %d allocated to memory block %d\n",i+1,first_fit_index+1);
      }       else
{
         printf("Process %d cannot be allocated\n",i+1);
      }    }    return
0;
}
```

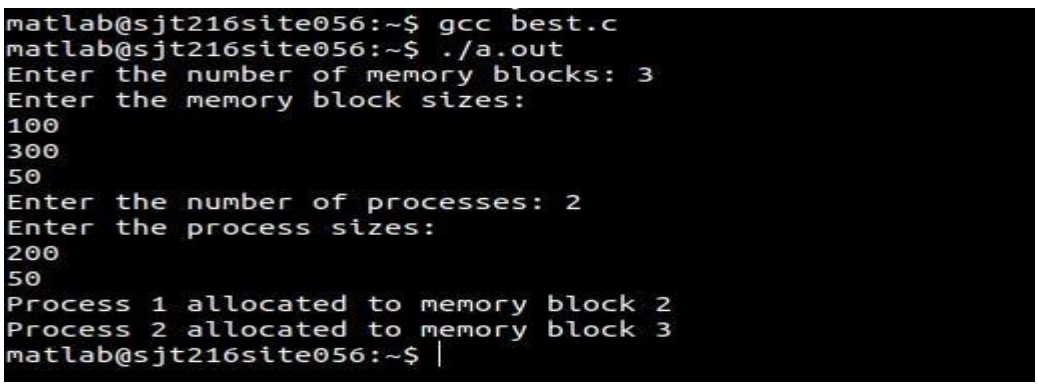
OUTPUT:



BEST FIT CODE:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 100
int main() {
   int memory[MAX_SIZE], n, process_size[MAX_SIZE], m, i, j, best_fit_index;
printf("Enter the number of memory blocks: ");
scanf("%d",&n);
   printf("Enter the memory block sizes: \n");
for(i=0;i<n;i++) {
scanf("%d",&memory[i]);
   }
   printf("Enter the number of processes: ");
scanf("%d",&m);
   printf("Enter the process sizes: \n");
for(i=0;i<m;i++) {
      scanf("%d",&process_size[i]);
   }
   for(i=0;i<m;i++) {        best_fit_index=-
1;       for(j=0;j<n;j++) {
if(memory[j] >= process_size[i]) {
if (best_fit_index==-1) {
                best_fit_index=j;
            }
            if(memory[j] < memory[best_fit_index]) {
                best_fit_index=j;
            }
        }
      }
      if(best_fit_index !=-1) {
         memory[best_fit_index] -= process_size[i];
         printf("Process %d allocated to memory block %d\n",i+1,best_fit_index+1);
      }       else
{
         printf("Process %d cannot be allocated\n",i+1);
      }   }    return
0;
}
```

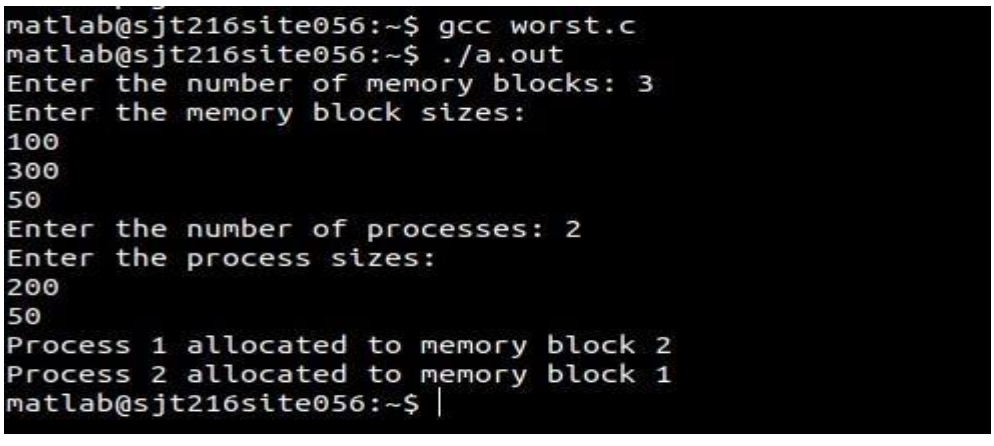
OUTPUT:



WORST FIT CODE:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_SIZE 100
int main() {
   int memory[MAX_SIZE], n, process_size[MAX_SIZE], m, i, j, worst_fit_index;
printf("Enter the number of memory blocks: ");    scanf("%d",&n);
   printf("Enter the memory block sizes: \n");
for(i=0;i<n;i++) {
scanf("%d",&memory[i]);
   }
   printf("Enter the number of processes: ");
scanf("%d",&m);
   printf("Enter the process sizes: \n");
for(i=0;i<m;i++) {
      scanf("%d",&process_size[i]);
   }
   for(i=0;i<m;i++) {
worst_fit_index=-1;       for(j=0;j<n;j++)
{
         if(memory[j] >= process_size[i]) {
            if(memory[j]==-1 || memory[j] > memory[worst_fit_index]) {
worst_fit_index=j;
            }
         }
      }
      if(worst_fit_index !=-1) {
         memory[worst_fit_index] -= process_size[i];
         printf("Process %d allocated to memory block %d\n",i+1,worst_fit_index+1);
      }       else
{
         printf("Process %d cannot be allocated\n",i+1);
      }    }    return
0;
}
```

OUTPUT:



QUESTION-2 FIFO CODE:

```
#include<stdio.h> int
main() {
int n,m,i,j,k,a,flag=0,pos,hits=0,flag1=0,pos1=0;
int fr[30][10],p[30]; printf("Enter the no of pages
:"); scanf("%d",&n);
printf("Enter the no of frames in memory:");
scanf("%d",&m);
printf("\nEnter the page numbers :\n
"); for(i=0;i<n;i++) {
scanf("%d",&p[i]); }
for(i=0;i<n;i++) {
for(j=0;j<m;j++)
 {  fr[i][j]=0;
 } }
for(i=0;i<n;i++)
{
flag=0;flag1=0;  if(i>0)
{
 a=i;
 for(k=0;k<m;k++)
 {
 fr[i][k]=fr[a-1][k];
 }  }
for(j=0;j<m;j++)
 {
 if(p[i]==fr[i][j])
 { flag=1; hits++;
}
 }
 if(flag==0)  {
for(k=0;k<m;k++)
 {  if(fr[i][k]==0)  {
pos=k;  flag1=1;
break;  }  }  if(flag1==1)
{  fr[i][pos]=p[i];
pos1=pos+1;
```

```
if(pos1>=m) pos1=0;
 }          else          {
fr[i][pos1]=p[i];
pos1=pos1+1;
if(pos1>=m) pos1=0;
 } } else
continue; } for(i=0;i<n;i++)
{ for(j=0;j<m;j++)
{
 printf("%d ",fr[i][j]);
}
printf("\n");
}
printf("\n\nno of page hits is:%d",hits);
printf("\nno of page faults is:%d\n",n-hits);
return 0;
}
```

OUTPUT:

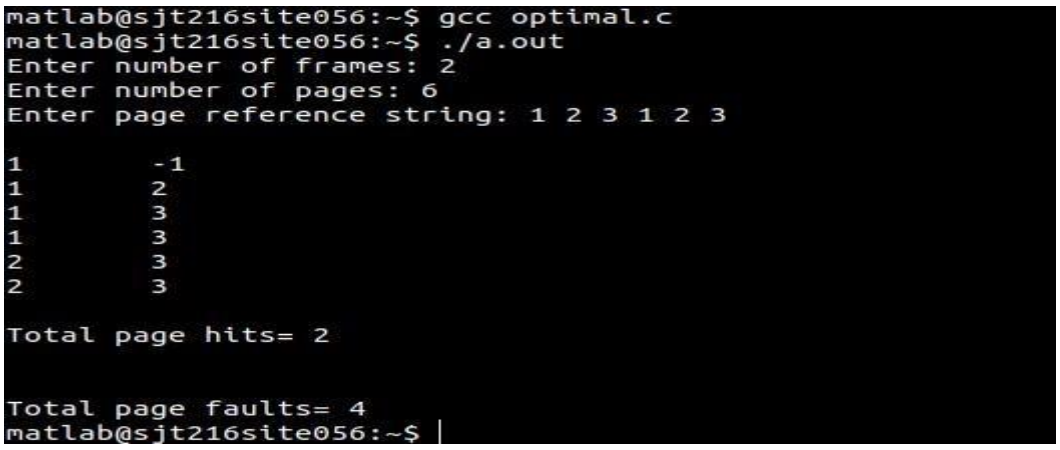
OPTIMAL CODE:

```
#include<stdio.h> int
main() {
   int nof, nop, frames[10], pages[30], temp[10], f1, f2, f3, i, j, k, pos, max,
faults=0;
```

```
   printf("Enter number of frames: ");
scanf("%d",&nof);
   printf("Enter number of pages: ");
scanf("%d",&nop);
   printf("Enter page reference string: ");
for(i=0;i<nop;++i) {
      scanf("%d",&pages[i]);
   }
   for(i=0;i<nof;++i) {
frames[i]=-1;
   }
   for(i=0;i<nop;++i) {
f1=f2=0;         for(j=0;j<nof;++j)
{          if(frames[j]==pages[i]) {
f1=f2=1;             break;
         }
      }
      if(f1==0) {
for(j=0;j<nof;++j) {
if(frames[j]==-1) {                faults++;
frames[j]=pages[i];
f2=1;                break;
           }
         }
      }       if(f2==0) {          f3=0;
for(j=0;j<nof;++j)                     {
temp[j]=-1;
for(k=i+1;k<nop;++k) {
               if(frames[j]==pages[k]) { temp[j]=k;
                        break;
               }
            }
         }
         for(j=0;j<nof;++j)          {
if(temp[j]==-1)              {          pos=j;
   f3=1;
               break;
```

```
                }
            }
if(f3==0) {
max=temp[0];
pos=0;
            for(j=1;j<nof;++j) {
if(temp[j]>max) {
max=temp[j];                    pos=j;
                }
            }
        }
        frames[pos]=pages[i];
faults++;       }       printf("\n");
for(j=0;j<nof;++j) {
printf("%d\t",frames[j]);
     }
  }
  printf("\n\nTotal page hits= %d\n",nop-faults);
printf("\n\nTotal page faults= %d\n", faults);
  return 0; }
```

OUTPUT:



LRU CODE:

```
#include<stdio.h> int
pa[30]; int
fr[30][10]; int i=0; int
nf,np; int getpos()
{
int j,k; int min,ret; int
```

```
temp[10][2]; for(j=0;j<nf;j++)
temp[j][0]=fr[i-1][j];
for(k=0;k<nf;k++)
for(j=0;j<i;j++)
if(temp[k][0]==pa[j])
 {
temp[k][1]=j;       }
min=temp[0][1];
ret=0;
for(j=0;j<nf;j++)   {
if(temp[j][1]<min)
{   min=temp[j][1];
ret=j;  }
}
return ret; }
void main()
{
int j;
int pos=0,temp,hits=0,flag=0,flag1=0;
printf("Enter number of pages");
scanf("%d",&np); printf("Enter frame
size"); scanf("%d",&nf); printf("Enter
all pages\n");      for(i=0;i<np;i++)
scanf("%d",&pa[i]);   for(i=0;i<np;i++)
for(j=0;j<nf;j++)                  fr[i][j]=0;
for(i=0;i<np;i++) {   flag=0;   flag1=0;
if(i>0) { temp=i;  for(j=0;j<nf;j++)
fr[i][j]=fr[temp-1][j];
 }
for(j=0;j<nf;j++)
{
if(fr[i][j]==pa[i])  {
hits++;  flag=1;
}
}
 if(flag==0)  {
for(j=0;j<nf;j++)
 {
```

```
if(fr[i][j]==0)
{  flag1=1;
 } }  if(flag1==1)
{
fr[i][pos]=pa[i];
pos+=1;
}
 else if(flag1==0)
{   pos=getpos();
fr[i][pos]=pa[i];
 }
 } }
printf("\nLEAST    RECENTLY    USED          ALGORITH");
printf("\nFrame1 frame2       frame3\n"); for(i=0;i<np;i++) {
 for(j=0;j<nf;j++)  printf("%d\t",fr[i][j]);
printf("\n");
}
printf("\nNO of page hits:%d\n",hits);
printf("\nNO of page faults:%d\n",np-hits);
}
```

OUTPUT: