1. Write C programs to simulate the following CPU Scheduling algorithms

a) FCFS

Programs:

```
#include<stdio.h>

#include<stdlib.h>


int main()

{

int n,i,b[20],w[20],t[20];

float aw,at;

printf("enter no of processes");

scanf("%d",&n);

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

{

printf("enter bust time for process %d",i+1);

scanf("%d",&b[i]);

}

w[0]=0;

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

{

w[i+1]=w[i]+b[i];
t[i]=w[i]+b[i];
}
printf("process     busttime     wating      tat\n");
for(i=0;i<n;i++)
{
printf(" p%d        %d          %d         %d\n",i,b[i],w[i],t[i]);
}
for(i=0;i<n;i++)
{
aw+=w[i];
at+=t[i];
}
aw=aw/n;
at=at/n;
printf("avg wating time is %f\n",aw);
printf("avg tat time is %f\n",at);
return 0;
}
```

Output:

Output 1:

```
enter no of processes3
enter bust time for process 13
enter bust time for process 26
enter bust time for process 32
process   busttime  wating    tat
p0          3          0      3
p1          6          3      9
p2          2          9     11
avg wating time is 4.000000
avg tat time is 7.666667
```

Output 2:

```
enter no of processes5
enter bust time for process 14
enter bust time for process 26
enter bust time for process 33
enter bust time for process 47
enter bust time for process 51
process   busttime  wating    tat
p0       4            0        4
p1        6            4      10
p2       3            10      13
p3        7           13      20
p4      1            20       21
avg wating time is 9.400000
avg tat time is 13.600000
```

b)SJF

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int n,b[10],w[10],t[10],i,h[10],temp=0,th=0,j;
float aw=0,at=0;
printf("enter number of processes\n");
scanf("%d",&n);

for(i=0;i<n;i++)
{
printf("enter the bust time of p%d\n",i);
scanf("%d",&b[i]);
h[i]=i;
}
for(i=0;i<n;i++)
{
```

```
for(j=i+1;j<n;j++)
{
if(b[i]>b[j])
{
temp=b[i];
b[i]=b[j];
b[j]=temp;
th=h[i];
h[i]=h[j];
h[j]=th;
}
}
}
w[0]=0;
printf("process      bust time        wating time        turn around
time\n");
for(i=0;i<n;i++)
{
w[i+1]=w[i]+b[i];
t[i]=b[i]+w[i];

printf("p%d          %d              %d                  %d
\n",h[i],b[i],w[i],t[i]);
```

```
}

for(i=0;i<n;i++)
{
aw+=w[i];
at+=t[i];
}
aw=aw/n;
at=at/n;
printf("avg wating time is %f\n",aw);
printf("avg tat time is %f\n",at);
return 0;
}
```

Output:

Output 1:

```
enter number of processes
4
enter the bust time of p0
4
enter the bust time of p1
2
enter the bust time of p2
```

```
6
enter the bust time of p3
1
process      bust time        wating time       turn around time
p3                  1              0                      1
p1                   2              1                     3
p0                   4              3                       7
p2                   6              7                      13
avg wating time is 2.750000
avg tat time is 6.000000
```

Output 2:

```
enter number of processes
5
enter the bust time of p0
4
enter the bust time of p1
6
enter the bust time of p2
1
enter the bust time of p3
7
```

```
enter the bust time of p4
5
process      bust time        wating time       turn around time
p2                  1              0                      1
p0                  4              1                      5
p4                  5              5                      10
p1                  6              10                     16
p3                  7              16                     23
avg wating time is 6.400000
avg tat time is 11.000000
```

c)round robbin

Program:

```
#include<stdio.h>
main()
{
int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
float awt=0,att=0,temp=0;
clrscr();
printf("Enter the no of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
```

```
{
printf("\nEnter Burst Time for process %d -- ", i+1);
scanf("%d",&bu[i]);
ct[i]=bu[i];
}
printf("\nEnter the size of time slice -- ");
scanf("%d",&t);
max=bu[0];
for(i=1;i<n;i++)
if(max<bu[i])
max=bu[i];
for(j=0;j<(max/t)+1;j++)
for(i=0;i<n;i++)
if(bu[i]!=0)
if(bu[i]<=t) {
tat[i]=temp+bu[i];
temp=temp+bu[i];
bu[i]=0;
}
else {
bu[i]=bu[i]-t;
temp=temp+t;
}
for(i=0;i<n;i++){
wa[i]=tat[i]-
ct[i]; att+=tat[i];
awt+=wa[i];}
printf("\nThe Average Turnaround time is -- %f",att/n);
printf("\nThe Average Waiting time is -- %f ",awt/n);
printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND
TIME\n");
for(i=0;i<n;i++)
printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);
getch();}
```

Output:

Output 1:

```
Enter the no of processes – 3
Enter Burst Time for process 1 – 24
Enter Burst Time for process 2 -- 3
Enter Burst Time for process 3 – 3
Enter the size of time slice – 3
```

OUTPUT:

```
PROCESS      BURST TIME   WAITING TIME   TURNAROUNDTIME
     1            24              6                30
```

```
      2             3                 4                   7

      3             3                 7                   10
The Average Turnaround time is – 15.666667
 The Average Waiting time is ------------ 5.666667
```

d)Priority

i)without condition

Program:

```
#include<stdio.h>
main()
{
int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp; float wtavg,
tatavg;
clrscr();
printf("Enter the number of processes --- ");
scanf("%d",&n);
for(i=0;i<n;i++){
p[i] = i;
printf("Enter the Burst Time & Priority of Process %d --- ",i); scanf("%d
%d",&bt[i], &pri[i]);
}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(pri[i] > pri[k]){
temp=p[i];
p[i]=p[k];
p[k]=temp;
temp=bt[i];
bt[i]=bt[k];
bt[k]=temp;
temp=pri[i];
pri[i]=pri[k];
pri[k]=temp;
}
wtavg = wt[0] = 0;
tatavg = tat[0] = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] + bt[i-1];
tat[i] = tat[i-1] + bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
```

```
printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING
TIME\tTURNAROUND

TIME");

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

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

printf("\nAverage Waiting Time is --- %f",wtavg/n); printf("\nAverage

Turnaround Time is --- %f",tatavg/n);

getch();}
```

Output:

```
Enter the number of processes -- 5

Enter the Burst Time & Priority of Process 0 --- 10 3

Enter the Burst Time & Priority of Process 1 --- 1 1

Enter the Burst Time & Priority of Process 2 --- 2 4

Enter the Burst Time & Priority of Process 3 --- 1 5

Enter the Burst Time & Priority of Process 4 --- 5 2

PROCESS PRIORITY  BURST TIME   WAITING TIME  TURNAROUND TIME

1            1            1                 0                 1

4             2           5                 1                 6

0            3             10               6                  16

2            4             2                16                18

3 5 1 18 19

Average Waiting Time is --- 8.200000
Average Turnaround Time is----------------- 12.000000
```

3) Write a C program to simulate Bankers Algorithm for Deadlock Avoidance and Prevention

Program:

```
#include<stdio.h>
void main()
{
int alloc[10][10],max[10][10];
int avail[10],work[10],total[10],need[10][10];
int i,j,n,m;
int count=0,c=0;
char finish[10];
printf("Enter the no.pf processes and resources:");
scanf("%d%d",&n,&m);
for(i=0;i<=n;i++)
finish[i]='n';
printf("Enter the Max matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&max[i][j]);
printf("Enter the allocation matrix:\n");
```

```
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&alloc[i][j]);
printf("Enter total instances of Resources:");
for(i=0;i<m;i++)
scanf("%d",&total[i]);
for(i=0;i<m;i++)
avail[i]=0;
for(i=0;i<n;i++)
for(j=0;j<m;j++)
avail[j]+=alloc[i][j];
for(i=0;i<m;i++)
work[i]=avail[i];
for(j=0;j<m;j++)
work[j]=total[j]-work[j];
for(i=0;i<n;i++)
for(j=0;j<m;j++)
need[i][j]=max[i][j]-alloc[i][j];
A:
for(i=0;i<n;i++)
{
c=0;
for(j=0;j<m;j++)

if((need[i][j]<=work[j])&&(finish[i]=='n'))

c++;

if(c==m)

{

printf("All the resources can be allocated to Process %d",i);

printf("\n\nAvailable resources are:");

for(j=0;j<m;j++)

{

work[j]+=alloc[i][j];

printf("%4d",work[j]);

}

printf("\n");

finish[i]='y';

printf("\nProcess %d executed: %c \n",i,finish[i]);

count++;

}

}

if(count!=n)

goto A;

else

printf("\n System is in safe mode");
printf("\n The given state is safe state");
}
```

Output:

```
Enter the no. of processes and resources: 4 3

Enter the claim matrix:

3 2 2

6 1 3

3 1 4

4 2 2

Enter the allocation matrix:

1 0 0

6 1 2

2 1 1

0 0 2

Resource vector:9 3 6

All the resources can be allocated to Process 2

Available resources are: 6 2 3

Process 2 executed?:y

All the resources can be allocated to Process 3 Available resources

are: 8 3 4

Process 3 executed?:y

All the resources can be allocated to Process 4 Available resources
```

```
are: 8 3 6

Process 4 executed?:y

All the resources can be allocated to Process 1

Available resources are: 9 3 6

Process 1 executed?:y

System is in safe mode

The given state is safe state
```

4. Write a C program to implement the Producer – Consumer problem using semaphores using UNIX/LINUX systemcalls.

Program:

```
#include<stdio.h>

#include<stdlib.h>

int mutex =1,full=0,empty=3,x=0;

int main()

{

int n;

void producer();

void consumer();

int wait(int);

int signal(int);

printf("\n1.Producer \n 2.consumer \n 3.exit");
```

```
while(1)

{

printf("\n Enter your choice");

scanf("%d",&n);

switch(n)

{

case 1:

if((mutex==1)&&(empty!=0))

producer();

else

        printf("\n Buffer is full");

break;

case 2:

if((mutex==1)&&(full!=0))

consumer();

else

printf("\nBuffer is empty!");

break;

case 3:

exit(0);

break;

}
}


}
int wait(int s)
{
return(--s);
}
int signal(int s)
{
return(++s);
}
void producer()
{
mutex=wait(mutex);
full=signal(full);
empty=wait(empty);
x++;
printf("\n Producer produces items : %d",x);
mutex=signal(mutex);
}
void consumer()
```

```
{
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\nConsumer consumes items : %d",x);
x--;
mutex=signal(mutex);
}
```

Output:

```
1. Produce 2. Consume 3. Exit
Enter your choice: 2
Buffer is Empty
1. Produce 2. Consume 3. Exit
Enter your choice: 1
Enter the value: 100
1. Produce 2. Consume 3. Exit
Enter your choice: 2
The consumed value is 100
1. Produce 2. Consume 3. Exit
Enter your choice: 3
```

5) Write C programs to illustrate the following IPC mechanisms

a) Pipes

Program:

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
#define MSG_LEN 64
int main()
{
int result;
int fd[2];
char message[MSG_LEN];
char recvd_msg[MSG_LEN];
result=pipe(fd);
printf("%d\n",result);
strncpy(message,"Linux world!! ",MSG_LEN);
result=write(fd[1],message,strlen(message));
printf("%d\n",result);
if(result<0)
{
```

```
perror("write");
exit(2);
}
strncpy(message,"understanding",MSG_LEN);
result=write(fd[1],message,strlen(message));
printf("%d\n",result);
if(result<0)
{
perror("write");
exit(2);
}
strncpy(message," concepts of",MSG_LEN);
result=write(fd[1],message,strlen(message));
printf("%d\n",result);
if(result<0)
{
perror("write");
exit(2);
}
strncpy(message," pipes ",MSG_LEN);
result=write(fd[1],message,strlen(message));
printf("%d\n",result);
```

```
if(result<0)
{
perror("write");
exit(2);
}
result=read(fd[0],recvd_msg,MSG_LEN);
if(result<0)
{
perror("read");
exit(3);
}
printf("%s\n",recvd_msg);
}
```

Program:

```
0
14
13
12
7
Linux World!! Understanding concepts of pipes
```

b) FIFOs

Program:

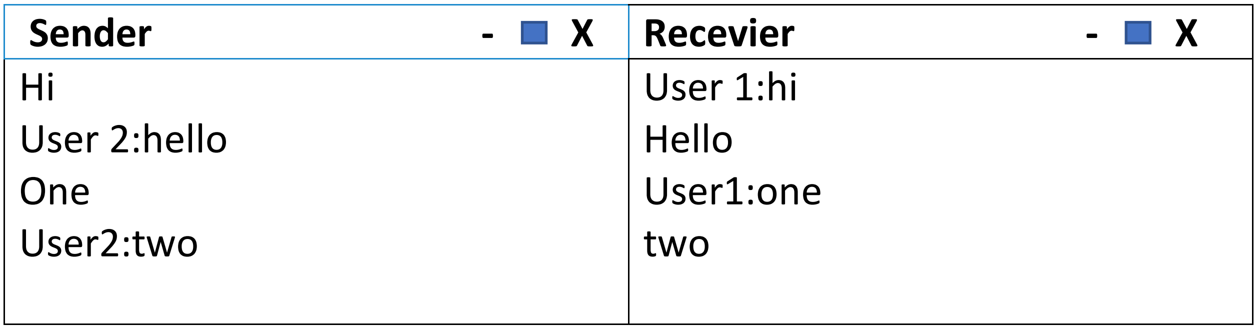
Sender program:

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
int main()
{
int fd;
char *myfifo1="/tmp/myfifo1";
mkfifo(myfifo1,0666);
char arr1[80],arr2[80];
while(1)
{
fd=open(myfifo1,O_WRONLY);
fgets(arr2,80,stdin);
write(fd,arr2,strlen(arr2)+1);
close(fd);
fd=open(myfifo1,O_RDONLY);
read(fd,arr1,sizeof(arr1));
printf("User2: %s\n",arr1);
close(fd);
}
return 0;
}
```

Recevier Program:

```
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<string.h>
int main()
{
int fd;
char *myfifo1="/tmp/myfifo1";
mkfifo(myfifo1,0666);
char arr1[80],arr2[80];
while(1)
{
fd=open(myfifo1,O_RDONLY);
read(fd,arr1,sizeof(arr1));
printf("User1: %s\n",arr1);
close(fd);
fd=open(myfifo1,O_WRONLY);

fgets(arr2,80,stdin);

write(fd,arr2,strlen(arr2)+1);

close(fd);

}

return 0;

}
```

Output:



Sender - ■ **X**	**Recevier** - ■ **X**
Hi User 2:hello One User2:two	User 1:hi Hello User1:one two

c)MessageQueues

Program:

Sender Program:

```
#include<stdio.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<sys/types.h>
```

```
#define MAX 10
struct mesg_buffer{
long mesg_type;
char mesg_text[100];
}message;
int main()
{
key_t key;
int msgid;
key=ftok("progfile",65);
msgid=msgget(key,0666|IPC_CREAT);
message.mesg_type=1;
printf("Write Data : ");
while(1)
{
fgets(message.mesg_text,MAX,stdin);
msgsnd(msgid,&message,sizeof(message),0);
printf("%Datasend is : %s\n",message.mesg_text);
}
return 0;
}
```

Recevier Program:

```
#include<stdio.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<sys/types.h>
#define MAX 10
struct mesg_buffer{
long mesg_type;
char mesg_text[100];
}message;
int main()
{
key_t key;
int msgid;
key=ftok("progfile",65);
msgid=msgget(key,0666|IPC_CREAT);
while(1)
{
msgrcv(msgid,&message,sizeof(message),1,0);
printf("Data Recieved is : %s\n",message.mesg_text);
}
msgct(msgid,IPC_RMID,NULL);
return 0;
```

}

Output:

```
Sender                    - ■ X | Recevier                    - ■ X
Write Data:Hi                   | Data Recevied is: Hi
Datasend is:Hi                  |
Hello                           | Data Recevied is:Hello
Datasend is :Hello              |
OS                              | Data Recevied is:OS
Datasend is:OS                  |
```

d) Shared

Program:

Terminal 1 code:

```
#include<stdio.h>

#include<unistd.h>

#include<sys/shm.h>

#include<string.h>

int main()

{

int i;

void*shared_memory;

char buff[100];
int shmid;
shmid=shmget((key_t)2345,1024,0666|IPC_CREAT);
shared_memory=shmat(shmid,NULL,0);
printf("Process attacted at %p\n",shared_memory);
printf("enter some data to write to shared memory\n");
read(0,buff,100);
strcpy(shared_memory,buff);
printf("you wrote: %s\n",(char *)shared_memory);
}
```

Terminal 2 code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void*shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345,1024,0666|IPC_CREAT);
shared_memory=shmat(shmid,NULL,0);

printf("Process attacted at %p\n",shared_memory);

printf("Data read from the shared memory is :%s\n",(char
*)shared_memory);

}
```

Output:

Terminal 1 - ■ X	**Terminal 2 - ■ X**
Process attached at 0x7faa1643d000 Enter some data or write to shared memory Shalini You wrote:Shalini	Process attached at 0x7faa1643d000 Dta read from shared memory is: Shalini