Simulate the following CPU scheduling algorithms: (a) FCFS (b) SJF

a) <u>FCFS</u>

<u>PROGRAM:</u>

```
# include<stdio.h>
struct fcfs
{
int at,st,str,ft,tat,wt;
}p[50];


main()
{
int i,j,n;
float atrt=0,awt=0;
clrscr();
printf("\nEnter the number of processes:");
scanf("%d",&n);
printf("\nEnter the arrival times of the processes");
for(i=0;i<n;i++)
scanf("%d",&p[i].at);
printf("\nEnter the service times of the processes");
for(i=0;i<n;i++)
scanf("%d",&p[i].st);
p[0].str=p[0].at;
for(j=0;j<n;j++)
{
p[j].ft=p[j].str+p[j].st;
p[j+1].str=p[j].ft;
}
for(i=0;i<n;i++)
{
p[i].tat=p[i].ft-p[i].at;
atrt=atrt+p[i].tat;
p[i].wt=p[i].str-p[i].at;
```

```
awt=awt+p[i].wt;
}
printf("process\tAT\tST\tSTR\tFT\tTAT\tWT\n");
for(i=0;i<n;i++)
{
printf("p%d\t%d\t%d\t%d\t%d\t%d\t%d\n",i,p[i].at,p[i].st,p[i].st
r,p[i].ft,p[i].tat,p[i].wt);
}
atrt=atrt/n;
awt=awt/n;
printf("Average turn around time=%f",atrt);
printf("Average waiting time=%f",awt);
getch();
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter the number of processes:4

Enter the arrival times of the processes2
3
4
5

Enter the service times of the processes5
6
7
8
process AT      ST      STR     FT      TAT     WT
p0      2       5       2       7       5       0
p1      3       6       7       13      10      4
p2      4       7       13      20      16      9
p3      5       8       20      28      23      15
Average turn around time=13.500000Average waiting time=7.000000
--------------------------------
Process exited after 21.26 seconds with return value 0
Press any key to continue . . .
```

b) SJF

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int et[20],at[10],n,i,j,temp,st[10],ft[10],wt[10],ta[10];
int totwt=0,totta=0;
float awt,ata;
char pn[10][10],t[10];
clrscr();
printf("Enter the number of process:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter process name, arrival time & service time:");
flushall();
scanf("%s%d%d",pn[i],&at[i],&et[i]);
}
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(et[i]<et[j])
{
temp=at[i];
at[i]=at[j];
at[j]=temp;
temp=et[i];
et[i]=et[j];
et[j]=temp;
strcpy(t,pn[i]);
strcpy(pn[i],pn[j]);
strcpy(pn[j],t);
}
}
for(i=0;i<n;i++)
{
```

```
if(i==0)
     st[i]=at[i];
else
     st[i]=ft[i-1];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
totwt+=wt[i];
totta+=ta[i];
}
awt=(float)totwt/n;
ata=(float)totta/n;
printf("\nPname \t arrivaltime \t servicetime \t waitingtime \t
tatime");
for(i=0;i<n;i++)
printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],wt[i],t
a[i]);
printf("\nAverage waiting time is:%f",awt);
printf("\nAverage turnaroundtime is:%f",ata);
getch();
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter the number of process:3
Enter process name, arrival time & service time:1 0 6
Enter process name, arrival time & service time:2 1 5
Enter process name, arrival time & service time:3 2 5

Pname    arrivaltime    servicetime    waitingtime    tatime
2          1              5              0              5
3          2              5              4              9
1          0              6             11             17
Average waiting time is:5.000000
Average turnaroundtime is:10.333333
--------------------------------
Process exited after 31.89 seconds with return value 0
Press any key to continue . . .
```

2. Simulate the following CPU scheduling algorithms: (a) Priority (b) Round Robin

A)Priority

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int et[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];
int totwt=0,totta=0;
float awt,ata;
char pn[10][10],t[10];
clrscr();
printf("Enter the number of process:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter process name, arrival time, execution time &
priority:");
flushall();
scanf("%s%d%d%d",pn[i],&at[i],&et[i],&p[i]);
}
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(p[i]<p[j])
{
temp=p[i];
p[i]=p[j];
p[j]=temp;
temp=at[i];
at[i]=at[j];
at[j]=temp;
temp=et[i];
```

```
et[i]=et[j];
et[j]=temp;
strcpy(t,pn[i]);
strcpy(pn[i],pn[j]);
strcpy(pn[j],t);
}
}
for(i=0;i<n;i++)
{
if(i==0)
{
st[i]=at[i];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
else
{
st[i]=ft[i-1];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
totwt+=wt[i];
totta+=ta[i];
}
awt=(float)totwt/n;
ata=(float)totta/n;
printf("\nPname \t arrivaltime \t executiontime \t priority \t
waitingtime \t    tatime");
for(i=0;i<n;i++)
printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[
i],p[i],wt[i],ta[i]);
printf("\nAverage waiting time is:%f",awt);
printf("\nAverage turnaroundtime is:%f",ata);
getch();
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter the number of process:3
Enter process name, arrival time, execution time & priority:1 0 6 1
Enter process name, arrival time, execution time & priority:2 1 10 2
Enter process name, arrival time, execution time & priority:3 2 9 1

Pname    arrivaltime    executiontime   priority       waitingtime       tatime
1          0              6              1              0              6
3          2              9              1              4             13
2          1             10              2             14             24
Average waiting time is:6.000000
Average turnaroundtime is:14.333333
--------------------------------
Process exited after 38.82 seconds with return value 0
Press any key to continue . . .
```

(b) Round Robin

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
main()
{
int b[10], pno[10],ts,n,s[10],e[10],w[10],t[10],r[10];
int I,c=0,x=0;
float aw=0,at=0;
printf("Enter number of processes");
scanf("%d",&n);
for(i=0;i<n;i++)
pno[i]=i+1;
printf("Enter the time slice");
scanf("%d",&ts);
printf("Enter the burst time of each process");
for(i=0;i<n;i++)
scanf("%d",&b[i]);
s[0]=0;
x=0;
c=0;
for(i=0;i<n;i++)
{
if(b[i]<ts)
{
e[i]=x+b[i];
r[i]=0;
}
else
{
e[i]=ts+x;
r[i]=b[i]-ts;
}
x=e[i];
s[i+1]=e[i];
t[i]=e[i];
w[i]=s[i];
}
```

```
while(c>=0)
{
    for(i=0;i<n;i++)
    {
    if(r[i]!=0)
    {
    w[i]=w[i]+x-e[i];
    if(r[i]<ts)
    {
    e[i]=x+r[i];
    r[i]=0;
    }
    else
    {
    e[i]=x+ts;
    r[i]=r[i]-ts;
    }
    x=e[i];
    t[i]=e[i];
    }
    if(r[i]!=0)
    c++;
    }
    c--;
}
for(i=0;i<n;i++)
{
aw=aw+w[i];
at=at+t[i];
}
aw=aw/n;
at=at/n;
printf("Time slice=%d",ts);
printf("\n pno \t bt \t st \t et \t wt \t tat");
for(i=0;i<n;i++)
printf("\n%d\t%d\t%d\t%d\t%d\t%d",pno[i],b[i],s[i],e[i],w[i],t[i]);
printf("\n Average waiting time=%f",aw);
printf("\nAverage turn around time=%f",at);
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter number of processes:4
Enter the time slice:3
Enter the burst time of each process:2
3
4
5
Time slice=3
 pno      bt       st       et       wt       tat
1        2        0        2        0        2
2        3        2        5        2        5
3        4        5        12       8        12
4        5        8        14       9        14
 Average waiting time=4.750000
Average turn around time=8.250000
--------------------------------
Process exited after 14.11 seconds with return value 0
Press any key to continue . . .
```

3.Multiprogramming-Memory management-Implementation of fork (), wait (), exec() and exit (), System calls

Program:

```
#include<stdio.h>
#include<sys/types.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
	pid_t p;
	int i;
	p=fork();
	if(p==1)
	{
		printf("fork error");
		exit(-1);
	}
	else if(p==0)
	{
		for(i=0;i<5;i++)
		{
			execlp("ls","ls","-l",NULL);
			printf("child process\n");
		}
	}
	else
	{
		wait(0);
		for(i=0;i<5;i++)
		{
			printf("parent process\n");
		}
		exit(0);
	}
}
```

Output:

```
total 24
-rwxr-xr-x 1 runner6 runner6 16912 Nov 29 13:51 a.out
-rwxrwxrwx 1 root    root      697 Nov 29 13:51 main.c
parent process
parent process
parent process
parent process
parent process


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

4. Simulate the Multiprogramming with a fixed number of tasks (MFT)

PROGRAM:

```
#include<stdio.h>
#include<math.h>
main()
{
int np,nb,mm,bs,i,j,ps[100],nba[100],ifm[100],sb=0,flag=0;
float x;
clrscr();
printf("Enter the Memory size");
scanf("%d",&mm);
printf("Enter the no of Blocks");
scanf("%d",&nb);
printf("Enter the no of processes");
scanf("%d",&np);
bs=mm/nb;
for(i=1;(i<=np)&&(sb<nb);i++)
{
printf("Enter the size of p[%d]:",i);
scanf("%d",&ps[i]);
if(ps[i]<=bs)
nba[i]=1;
else
{
x=ps[i]/(float)bs;
nba[i]=(ceil)(x);
}
ifm[i]=nba[i]*bs-ps[i];
sb=sb+nba[i];
if(sb>nb)
{
i=i-1;
flag=1;
}
}
```

```
j=i;
printf("Process \t Size \t nba \t ifm \n");
for(i=1;i<j;i++)
    printf("%d \t %d \t %d \t %d \n", i, ps[i], nba[i], ifm[i]);
if(flag==1)
    printf("Memory space is unavailable");
getch();
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter the Memory size:800
Enter the no of Blocks:8
Enter the no of processes:4
Enter the size of p[1]:50
Enter the size of p[2]:100
Enter the size of p[3]:150
Enter the size of p[4]:200
Process Size    nba     ifm
1       50      1       50
2       100     1       0
3       150     2       50
4       200     2       0

--------------------------------
Process exited after 23.18 seconds with return value 0
Press any key to continue . . .
```

5. Simulate the Multiprogramming with a variable number of tasks (MVT)

PROGRAM:

```
#include<stdio.h>
main()
{
int mm,np,ps[100],rm[100],am=0,flag=0,i,j;
clrscr();
printf("Enter the memory size");
scanf("%d",&mm);
printf("enter no of processes");
scanf("%d",&np);
for(i=0;(i<np)&&(am<mm);i++)
{
printf("Enter the size of p[%d]:",i+1);
scanf("%d",&ps[i]);
am=am+ps[i];
if(am>=mm)
{
flag=1;
break;
}
rm[i]=mm-am;
}
j=i;
printf("Process \t size \t rm \n");
for(i=0;i<j;i++)
printf("%d \t %d \t %d \n ", i+1, ps[i] , rm[i]);
if(flag==1)
printf("memory is unavailable");
getch();
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter the memory size:500
enter no of processes:5
Enter the size of p[1]:50
Enter the size of p[2]:100
Enter the size of p[3]:150
Enter the size of p[4]:50
Enter the size of p[5]:100
Process size    rm
1       50      450
 2      100     350
 3      150     200
 4      50      150
 5      100     50

--------------------------------
Process exited after 24.2 seconds with return value 0
Press any key to continue . . .
```

9. Simulate the following File allocation strategies (a)Sequenced (b) Indexed (c) Linked

PROGRAM:

```
#include<stdio.h>
#include<string.h>
main()
{
int i,j,n,size[50],sblock[20],eblock[20];
char name[50];
printf("Enter no of files:");
scanf("%d",&n);
printf("enter file name and size and starting block:");
for(i=0;i<n;i++)
{
scanf(" %c %d %d",&name[i],&size[i],&sblock[i]);
}
for(i=0;i<n;i++)
{
eblock[i]=sblock[i]+size[i];
}
printf("file allocation table\n");
printf("name size startblock endblock\n");
for(i=0;i<n;i++)
{
printf("%c\t%d\t",name[i],size[i]);
printf("%d\t%d",sblock[i],eblock[i]);
printf("\n");
}
}
```

Output:

C:\Users\durga\OneDrive\Pictures\Documents\os.exe

```
Enter no of files:3
enter file name and size and starting block:a 3 1
b 5 7
c 8 20
file allocation table
name size startblock endblock
a       3       1       4
b       5       7       12
c       8       20      28

--------------------------------
Process exited after 22.7 seconds with return value 0
Press any key to continue . . .
```