C Program for First Come First Serve (FCFS)

```
#include<stdio.h>
main()
{
    int n,i,j,sum=0;
    int arrv[10],ser[10],start[10];
    int finish[10],wait[10],turn[10];
    float avgturn=0.0,avgwait=0.0;
    start[0]=0;

    printf("Enter the number of processes:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter the arriwal and service time of %d process:",i+1);
        scanf("%d%d",&arrv[i],&ser[i]);
    }
    for(i=0;i<n;i++)
    {
        sum=0;
        for(j=0;j<i;j++)
        sum=sum+ser[j];
        start[i]=sum;
    }
    for(i=0;i<n;i++)
```

```
    {
        finish[i]=ser[i]+start[i];
        wait[i]=start[i];
        turn[i]=ser[i]+wait[i];
    }
    for(i=0;i<n;i++)
    {
        avgwait+=wait[i];
        avgturn+=turn[i];
    }
    avgwait/=n;
    avgturn/=n;
    printf("\narraival service Start Finish Wait Turn\n");
    for(i=0;i<n;i++)
    printf("%d\t%d\t%d\t%d\t%d\t%d\n",arrv[i],ser[i],start[i],
    finish[i],wait[i],turn[i]);
    printf("\nAverage waiting time=%f",avgwait);
    printf("\nAverage turn around time=%f",avgturn);

}
```

Output:

```
File  Edit  View  Search  Terminal  Help
meghna@root:~/practical/os$ gcc fcfs.c -o fcfs
meghna@root:~/practical/os$ ./fcfs
Enter the number of processes:3
Enter the arriwal and service time of 1 process:3 2
Enter the arriwal and service time of 2 process:2 4
Enter the arriwal and service time of 3 process:5 3

arraival service Start Finish Wait Turn
3       2       0       2       0       2
2       4       2       6       2       6
5       3       6       9       6       9

Average waiting time=2.666667
Average turn around time=5.666667meghna@root:~/practical/os$
```

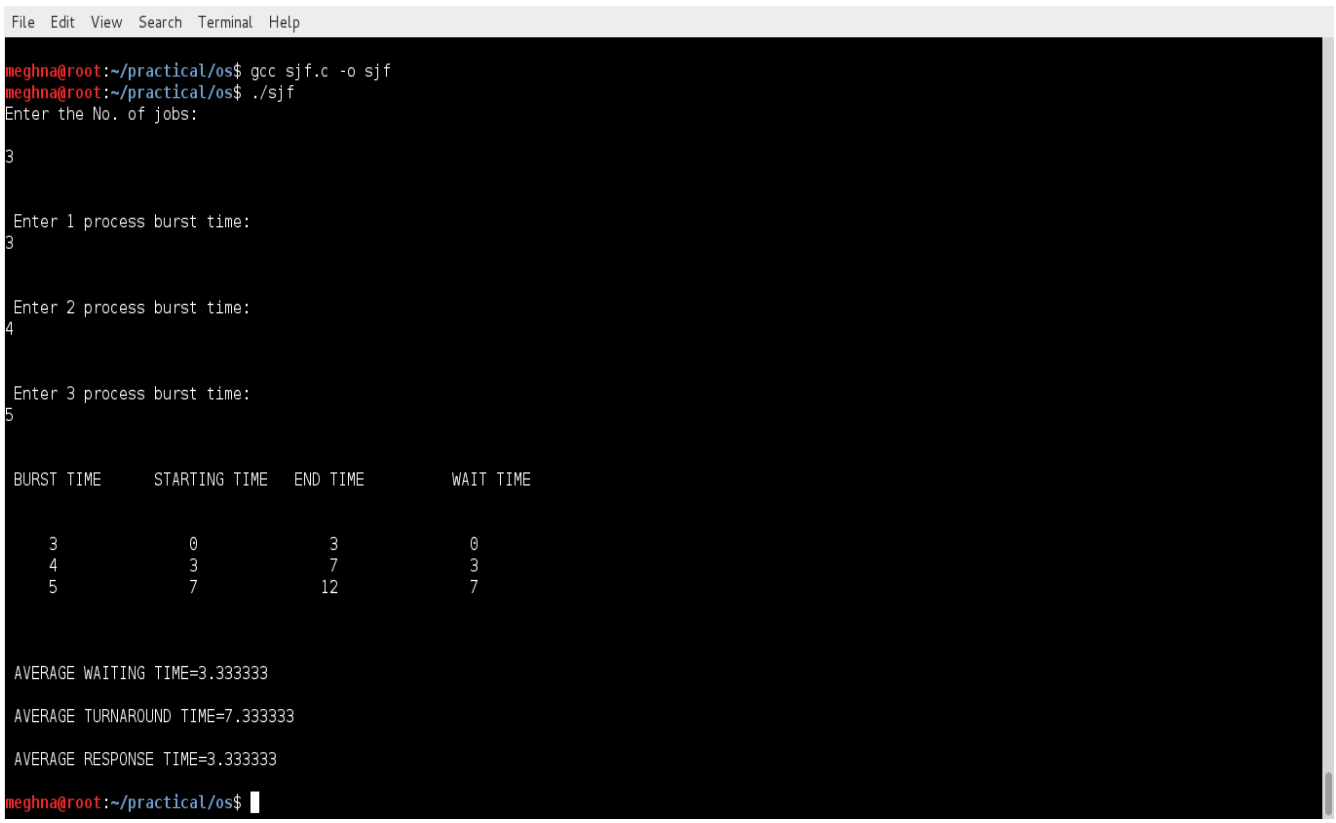
C Program for Shortest Job First Scheduling (SJF)

```
#include<stdio.h>



void main()
{
    int i,j,n,brust_time[10],start_time[10],end_time[10],wait_time[10],temp,tot;
    float avg;

    printf("Enter the No. of jobs:\n\n");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("\n \n Enter %d process burst time:\n",i);
        scanf("%d",&brust_time[i]);
    }

    for(i=1;i<=n;i++)
    {
        for(j=i+1;j<=n;j++)
        {
            if(brust_time[i]>brust_time[j])
            {
                temp=brust_time[i];
                brust_time[i]=brust_time[j];
```

```
            brust_time[j]=temp;
        }
    }

    if(i==1)
    {
        start_time[1]=0;
        end_time[1]=brust_time[1];
        wait_time[1]=0;
    }

    else
    {
        start_time[i]=end_time[i-1];
        end_time[i]=start_time[i]+brust_time[i];
        wait_time[i]=start_time[i];
    }
}
printf("\n\n BURST TIME \t STARTING TIME \t END TIME \t   WAIT TIME\n");
printf("\n");
for(i=1;i<=n;i++)
{
    printf("\n %5d %15d %15d %15d",brust_time[i],start_time[i],end_time[i],wait_time[i]);
}
printf("\n");
for(i=1,tot=0;i<=n;i++)
tot+=wait_time[i];
avg=(float)tot/n;
```

```
    printf("\n\n\n AVERAGE WAITING TIME=%f",avg);
    for(i=1,tot=0;i<=n;i++)
    tot+=end_time[i];
    avg=(float)tot/n;
    printf("\n\n AVERAGE TURNAROUND TIME=%f",avg);
    for(i=1,tot=0;i<=n;i++)
    tot+=start_time[i];
    avg=(float)tot/n;
    printf("\n\n AVERAGE RESPONSE TIME=%f\n\n",avg);

}
```

Output:



C Program for Shortest Remaining Time First (SRTF)

```
#include<stdio.h>
int main()
{
    int at[10],bt[10],rt[10],endTime,i,smallest;
    int remain=0,n,time,sum_wait=0,sum_turnaround=0;
    printf("Enter no of Processes : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter arrival time for Process P%d : ",i+1);
        scanf("%d",&at[i]);
        printf("Enter burst time for Process P%d : ",i+1);
        scanf("%d",&bt[i]);
        rt[i]=bt[i];
    }
    printf("\n\nProcess\t|Turnaround Time| Waiting Time\n\n");
    rt[9]=9999;
    for(time=0;remain!=n;time++)
    {
        smallest=9;
        for(i=0;i<n;i++)
        {
            if(at[i]<=time && rt[i]<rt[smallest] && rt[i]>0)
            {
```

```
            smallest=i;
        }
    }
    rt[smallest]--;
    if(rt[smallest]==0)
    {
        remain++;
        endTime=time+1;
                printf("\nP[%d]\t|\t%d\t|\t%d",smallest+1,endTime-at[smallest],endTime-bt[smallest]-
at[smallest]);
        sum_wait+=endTime-bt[smallest]-at[smallest];
        sum_turnaround+=endTime-at[smallest];
    }
  }
  printf("\n\nAverage waiting time = %f\n",sum_wait*1.0/n);
  printf("Average Turnaround time = %f",sum_turnaround*1.0/5);
  return 0;
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
meghna@root:~/practical/os$ gcc srtf.c -o srtf
meghna@root:~/practical/os$ ./srtf
Enter no of Processes : 3
Enter arrival time for Process P1 : 2
Enter burst time for Process P1 : 4
Enter arrival time for Process P2 : 1
Enter burst time for Process P2 : 3
Enter arrival time for Process P3 : 2
Enter burst time for Process P3 : 6


Process |Turnaround Time| Waiting Time


P[2]    |       3       |       0
P[1]    |       6       |       2
P[3]    |       12      |       6

Average waiting time = 2.666667
meghna@root:~/practical/os$
```

C Program for Priority Scheduling

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

   int x,n,p[10],pp[10],pt[10],w[10],t[10],awt,atat,i;
   printf("Enter the number of process : ");
   scanf("%d",&n);
   printf("\n Enter process : time priorities \n");
   for(i=0;i<n;i++)
    {
      printf("\nProcess no %d : ",i+1);
      scanf("%d  %d",&pt[i],&pp[i]);
      p[i]=i+1;
    }
  for(i=0;i<n-1;i++)
   {
     for(int j=i+1;j<n;j++)
      {
        if(pp[i]<pp[j])
         {
           x=pp[i];
           pp[i]=pp[j];
           pp[j]=x;
           x=pt[i];
```

```
        pt[i]=pt[j];
        pt[j]=x;
        x=p[i];
        p[i]=p[j];
        p[j]=x;
      }
   }
}
w[0]=0;
awt=0;
t[0]=pt[0];
atat=t[0];
for(i=1;i<n;i++)
 {
   w[i]=t[i-1];
   awt+=w[i];
   t[i]=w[i]+pt[i];
   atat+=t[i];
 }
printf("\n\n Job \t Burst Time \t Wait Time \t Turn Around Time   Priority \n");
for(i=0;i<n;i++)
  printf("\n %d \t\t %d  \t\t %d \t\t %d \t\t %d \n",p[i],pt[i],w[i],t[i],pp[i]);
awt/=n;
atat/=n;
printf("\n Average Wait Time : %d \n",awt);
printf("\n Average Turn Around Time : %d \n",atat);

}
```

Output:

```
File  Edit  View  Search  Terminal  Help
meghna@root:~/practical/os$ ./priority
Enter the number of process : 3

 Enter process : time priorities

Process no 1 : 3 2

Process no 2 : 4 1

Process no 3 : 5 3


 Job     Burst Time      Wait Time       Turn Around Time   Priority

 3              5              0              5              3

 1              3              5              8              2

 2              4              8              12             1

 Average Wait Time : 4

 Average Turn Around Time : 8
meghna@root:~/practical/os$
```

C Program For Round Robin Scheduling

```
#include<stdio.h>
struct process
{
    char na[20];
    int at,bt,ft,tat,rem;
    float ntat;
}Q[5],temp;
int rr[20],q,x,k;
main()
{
    int f,r,n,i,j,tt=0,qt,t,flag,wt=0;
    float awt=0,antat=0,atat=0;

    printf("Enter the no. of jobs:");
    scanf("%d",&n);
    for(r=0;r<n;r++)
    {
        printf("Enter process name,arrival time and burst time:\n");
        scanf("%s%d%d",Q[r].na,&Q[r].at,&Q[r].bt);
    }
    printf("Enter quantum:\n");
    scanf("%d",&qt);
    for(i=0;i<n;i++)
    {
```

```
    for(j=i+1;j<n;j++)
    {
        if(Q[i].at>Q[j].at)
        {
            temp=Q[i];
            Q[i]=Q[j];
            Q[j]=temp;
        }
    }
}
for(i=0;i<n;i++)
{
    Q[i].rem=Q[i].bt;
    Q[i].ft=0;
}
tt=0;
q=0;
rr[q]=0;
do
{
    for(j=0;j<n;j++)
    if(tt>=Q[j].at)
    {
        x=0;
        for(k=0;k<=q;k++)
        if(rr[k]==j)
        x++;
        if(x==0)
```

```
    {
        q++;
        rr[q]=j;
    }
}
if(q==0)
i=0;
if(Q[i].rem==0)
i++;
if(i>q)
i=(i-1)%q;
if(i<=q)
{
    if(Q[i].rem>0)
    {
        if(Q[i].rem<qt)
        {
            tt+=Q[i].rem;
            Q[i].rem=0;
        }
        else
        {
            tt+=qt;
            Q[i].rem-=qt;
        }
        Q[i].ft=tt;
    }
    i++;
```

```
    }
    flag=0;
    for(j=0;j<n;j++)
    if(Q[j].rem>0)
    flag++;
}while(flag!=0);

printf("\n\n\t\tROUND ROBIN ALGORITHM");
printf("\n");
printf("\nprocesses Arrival time burst time finish time tat wt ntat");
for(f=0;f<n;f++)
{
    wt=Q[f].ft-Q[f].bt-Q[f].at;
    Q[f].tat=Q[f].ft-Q[f].at;
    Q[f].ntat=(float)Q[f].tat/Q[f].bt;
    antat+=Q[f].ntat;
    atat+=Q[f].tat;
    awt+=wt;
    printf("\n\t%s\t%d\t\t%d\t%d\t%d\t%d %f",
    Q[f].na,Q[f].at,Q[f].bt,Q[f].ft,Q[f].tat,wt,Q[f].ntat);
}
antat/=n;
atat/=n;
awt/=n;
printf("\nAverage tat is %f",atat);
printf("\nAverage normalised tat is %f",antat);
printf("\n average waiting time is %f",awt);
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
meghna@root:~/practical/os$ gcc rr.c -o rr
meghna@root:~/practical/os$ ./rr
Enter the no. of jobs:3
Enter process name,arrival time and burst time:
first 3 4
Enter process name,arrival time and burst time:
second 5 3
Enter process name,arrival time and burst time:
third 4 2
Enter quantum:
2


                ROUND ROBIN ALGORITHM

processes Arrival time burst time finish time tat wt ntat
        first   3              4       4       1       -3 0.250000
        third   4              2       6       2       0 1.000000
        second  5              3       9       4       1 1.333333
Average tat is 2.333333
Average normalised tat is 0.861111
 average waiting time is -0.666667meghna@root:~/practical/os$
```

C Program for Page Replacement Algorithm

```
#include<stdio.h>
int fr[3];
void main()
{
void display();
int p[12]={2,3,2,1,5,2,4,5,3,2,5,2},i,j,fs[3];
int max,found=0,lg[3],index,k,l,flag1=0,flag2=0,pf=0,frsize=3;

for(i=0;i<3;i++)
{
fr[i]=-1;
}
for(j=0;j<12;j++)
{
flag1=0;
flag2=0;
for(i=0;i<3;i++)
{
if(fr[i]==p[j])
{
flag1=1;
flag2=1;
break;
}
}
```

```
if(flag1==0)
{
for(i=0;i<3;i++)
{
if(fr[i]==-1)
{
fr[i]=p[j];
flag2=1;
break;
}
}
}

if(flag2==0)
{
for(i=0;i<3;i++)
lg[i]=0;
for(i=0;i<frsize;i++)
{
for(k=j+1;k<12;k++)
{
if(fr[i]==p[k])
{
lg[i]=k-j;
break;
}
}
}
```

```
found=0;
for(i=0;i<frsize;i++)
{
if(lg[i]==0)
{
index=i;
found=1;
break;
}
}
if(found==0)
{
max=lg[0];
index=0;
for(i=1;i<frsize;i++)
{
if(max<lg[i])
{
max=lg[i];
index=i;
}
}
}
fr[index]=p[j];
pf++;
}
display();
}
```

```
printf("\n no of page faults:%d",pf);


}
void display()
{
int i;
printf("\n");
for(i=0;i<3;i++)
printf("\t%d",fr[i]);
}
```

output:

```
File  Edit  View  Search  Terminal  Help
meghna@root:~/practical/os$ gcc pagerep.c -o pagereplacement
meghna@root:~/practical/os$ ./pagereplacement

        2       -1      -1
        2       3       -1
        2       3       -1
        2       3       1
        2       3       5
        2       3       5
        4       3       5
        4       3       5
        4       3       5
        2       3       5
        2       3       5
        2       3       5
 no of page faults:3meghna@root:~/practical/os$
```

C program for disk scheduling

```
#include<stdio.h>
void main()
{
 int a[20],n,i,t=0;

 printf("Enter head pointer position:");
 scanf("%d",&a[0]);
 printf("\nEnter number of processes:");
 scanf("%d",&n);
 printf("\nEnter processes in request order:");
 for(i=1;i<=n;i++)
 {
  scanf("%d",&a[i]);
 }
 for(i=0;i<n;i++)
 {
  if(a[i]<a[i+1])
  t+=(a[i+1]-a[i]);
  else21
  t+=(a[i]-a[i+1]);
 }
 printf("\nProcessing order:");
 for(i=0;i<=n;i++)
 printf("\t%d",a[i]);
```

```
 printf("\nTotal Head Movement:%d",t);
 printf("\n");
}
```

Output:

```
File  Edit  View  Search  Terminal  Help
meghna@root:~/practical/os$ gcc diskscheduling.c -o diskscheduling
meghna@root:~/practical/os$ ./diskscheduling
Enter head pointer position:342

Enter number of processes:3

Enter processes in request order:2 3 5

Processing order:       342     2       3       5
Total Head Movement:343
meghna@root:~/practical/os$
```