Name: **Shangirne Kharbanda**

Registration Number: **20BAI1154**

OS LAB-4

1. Consider a system containing four processes P1, P2, P3 P4, and P5 with burst time 8, 7, 5,

2 and 10 time units (ms) arrive at same time. For the given Process, Arrival Time and Burst

time, construct the

Gantt Chart for

- **First Come First Serve**

- **Shortest Job First using (Non-Preemptive)**

- **Priority Scheduling (Non-Preemptive)**

- **Round Robin (Time Slice-2)**

The processes are allowed to compute the work based on the priority 4, 3 ,1,2,5,

respectively.

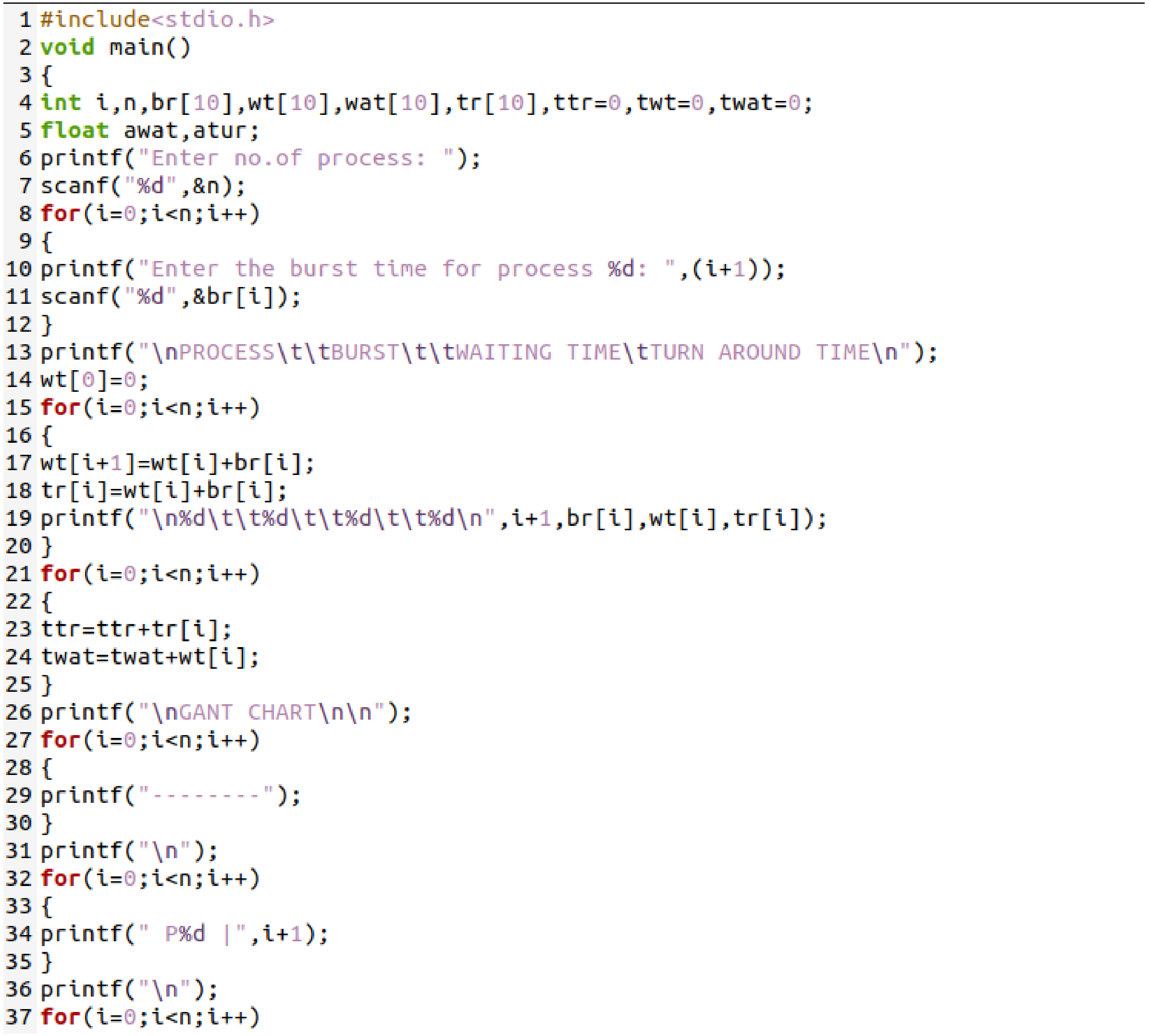
Calculate the average waiting time for the scheduling method and justify which Scheduling

method could be recommended ensuring higher performance of the system. Evaluate with

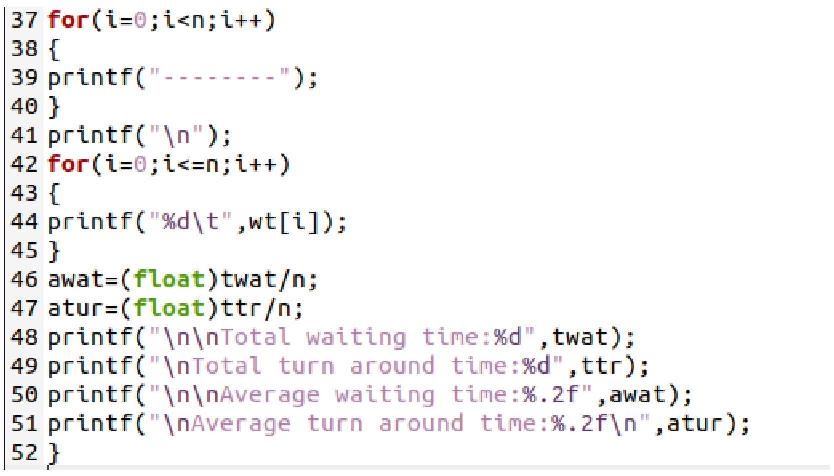
respect to Average waiting time and turn-around time.

<u>FIFS Scheduling</u>

Code:

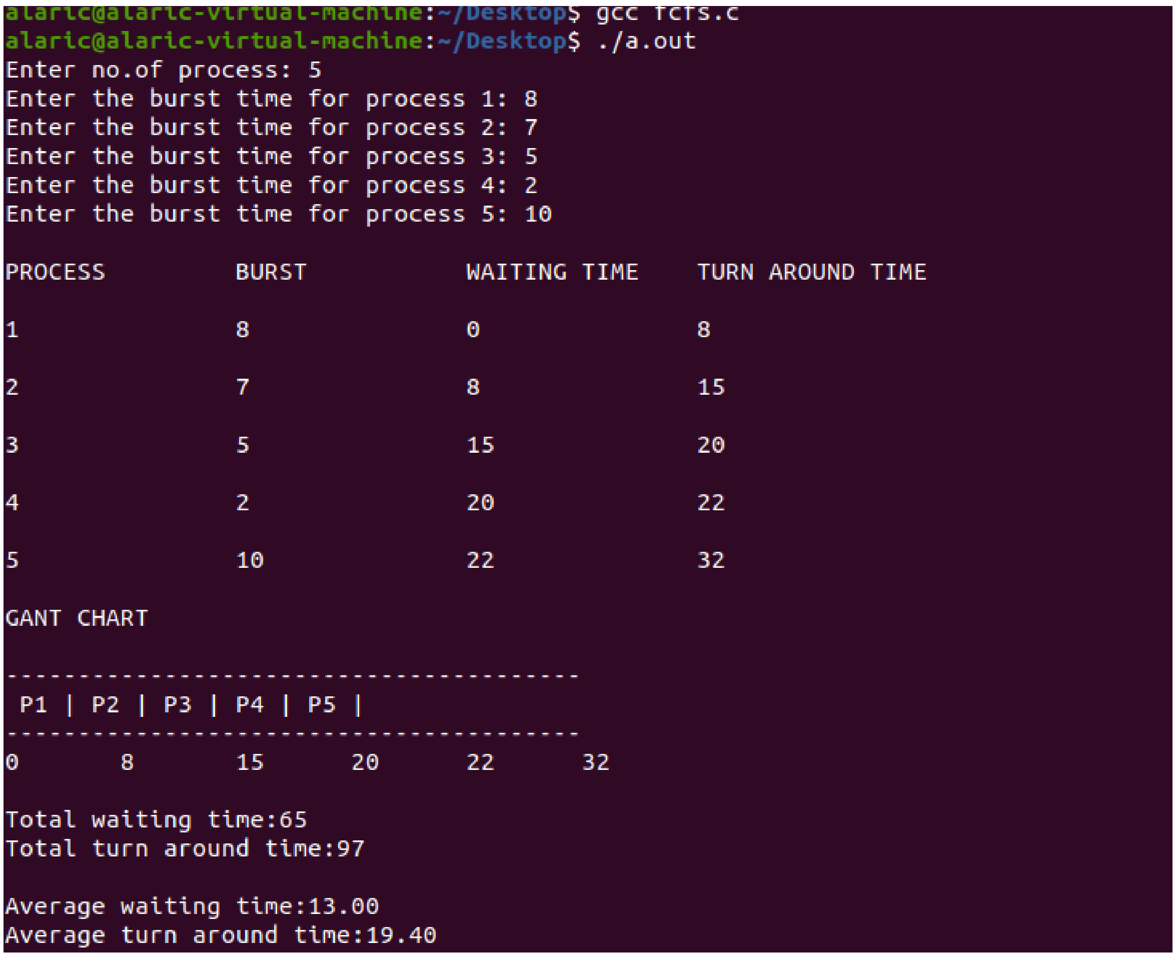


```c
#include<stdio.h>
void main()
{
int i,n,br[10],wt[10],wat[10],tr[10],ttr=0,twt=0,twat=0;
float awat,atur;
printf("Enter no.of process: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the burst time for process %d: ",(i+1));
scanf("%d",&br[i]);
}
printf("\nPROCESS\t\tBURST\t\tWAITING TIME\tTURN AROUND TIME\n");
wt[0]=0;
for(i=0;i<n;i++)
{
wt[i+1]=wt[i]+br[i];
tr[i]=wt[i]+br[i];
printf("\n%d\t\t%d\t\t%d\t\t%d\n",i+1,br[i],wt[i],tr[i]);
}
for(i=0;i<n;i++)
{
ttr=ttr+tr[i];
twat=twat+wt[i];
}
printf("\nGANT CHART\n\n");
for(i=0;i<n;i++)
{
printf("--------");
}
printf("\n");
for(i=0;i<n;i++)
{
printf(" P%d |",i+1);
}
printf("\n");
for(i=0;i<n;i++)
```



```c
for(i=0;i<n;i++)
{
printf("--------");
}
printf("\n");
for(i=0;i<=n;i++)
{
printf("%d\t",wt[i]);
}
awat=(float)twat/n;
atur=(float)ttr/n;
printf("\n\nTotal waiting time:%d",twat);
printf("\nTotal turn around time:%d",ttr);
printf("\n\nAverage waiting time:%.2f",awat);
printf("\nAverage turn around time:%.2f\n",atur);
}
```

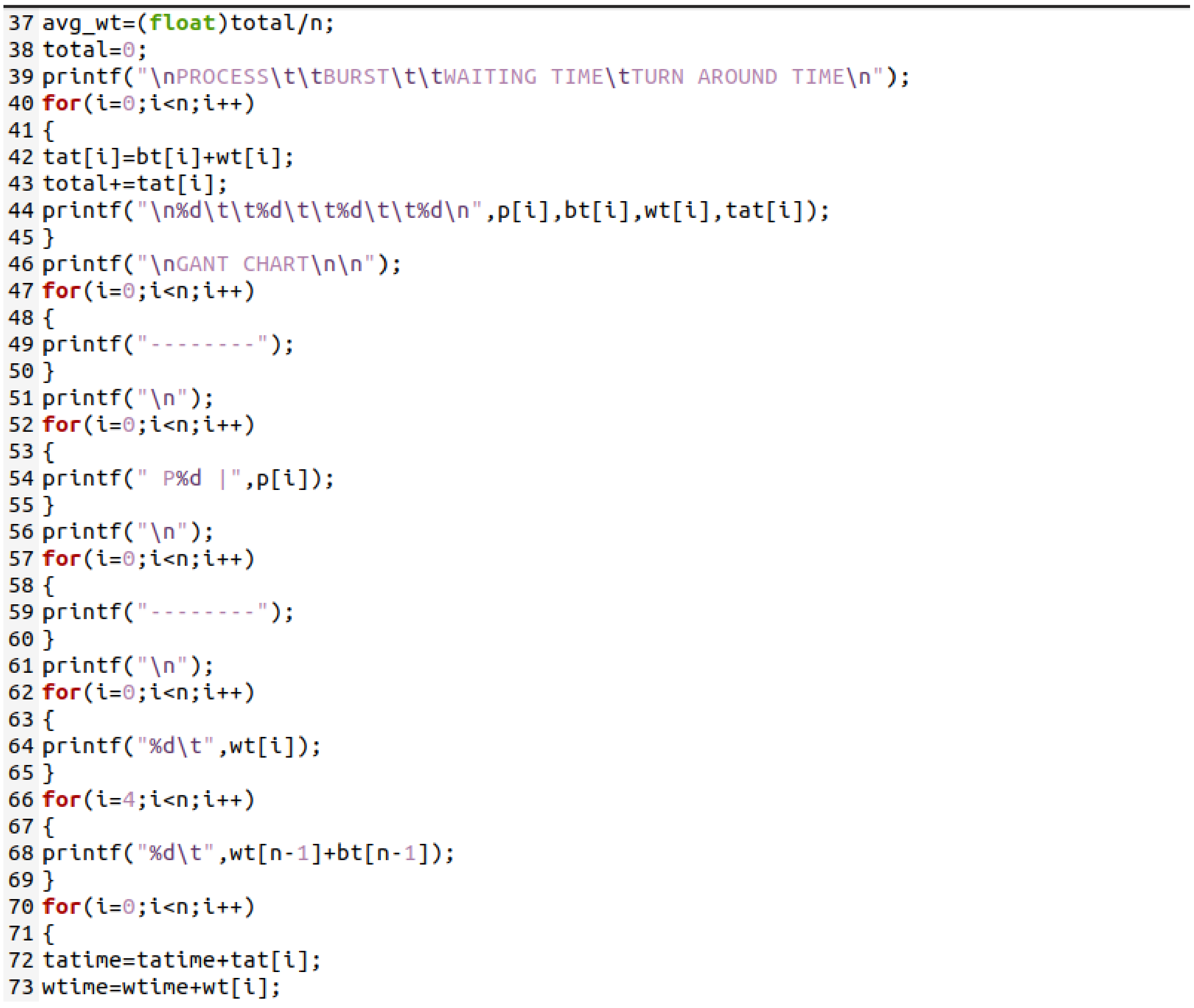
Output:


```
alaric@alaric-virtual-machine:~/Desktop$ gcc fcfs.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
Enter no.of process: 5
Enter the burst time for process 1: 8
Enter the burst time for process 2: 7
Enter the burst time for process 3: 5
Enter the burst time for process 4: 2
Enter the burst time for process 5: 10

PROCESS         BURST           WAITING TIME    TURN AROUND TIME

1               8               0               8

2               7               8               15

3               5               15              20

4               2               20              22

5               10              22              32

GANT CHART

----------------------------------------
 P1 | P2 | P3 | P4 | P5 |
----------------------------------------
0       8       15      20      22      32

Total waiting time:65
Total turn around time:97

Average waiting time:13.00
Average turn around time:19.40
```

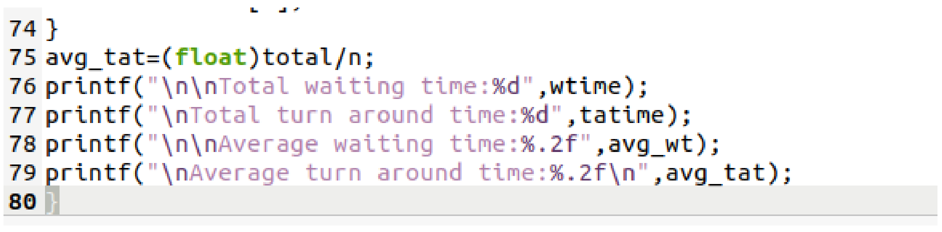
<u>SJF Scheduling:</u>

Code:

```c
#include<stdio.h>
int main()
{
int bt[10],p[10],wt[10],tat[10],i,j,n,total=0,pos,temp,wtime=0,tatime=0;
float avg_wt,avg_tat;
printf("Enter no. of process: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the burst time for process %d: ",(i+1));
scanf("%d",&bt[i]);
p[i]=i+1;
}
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(bt[j]<bt[pos])
pos=j;
}
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=(float)total/n;
```

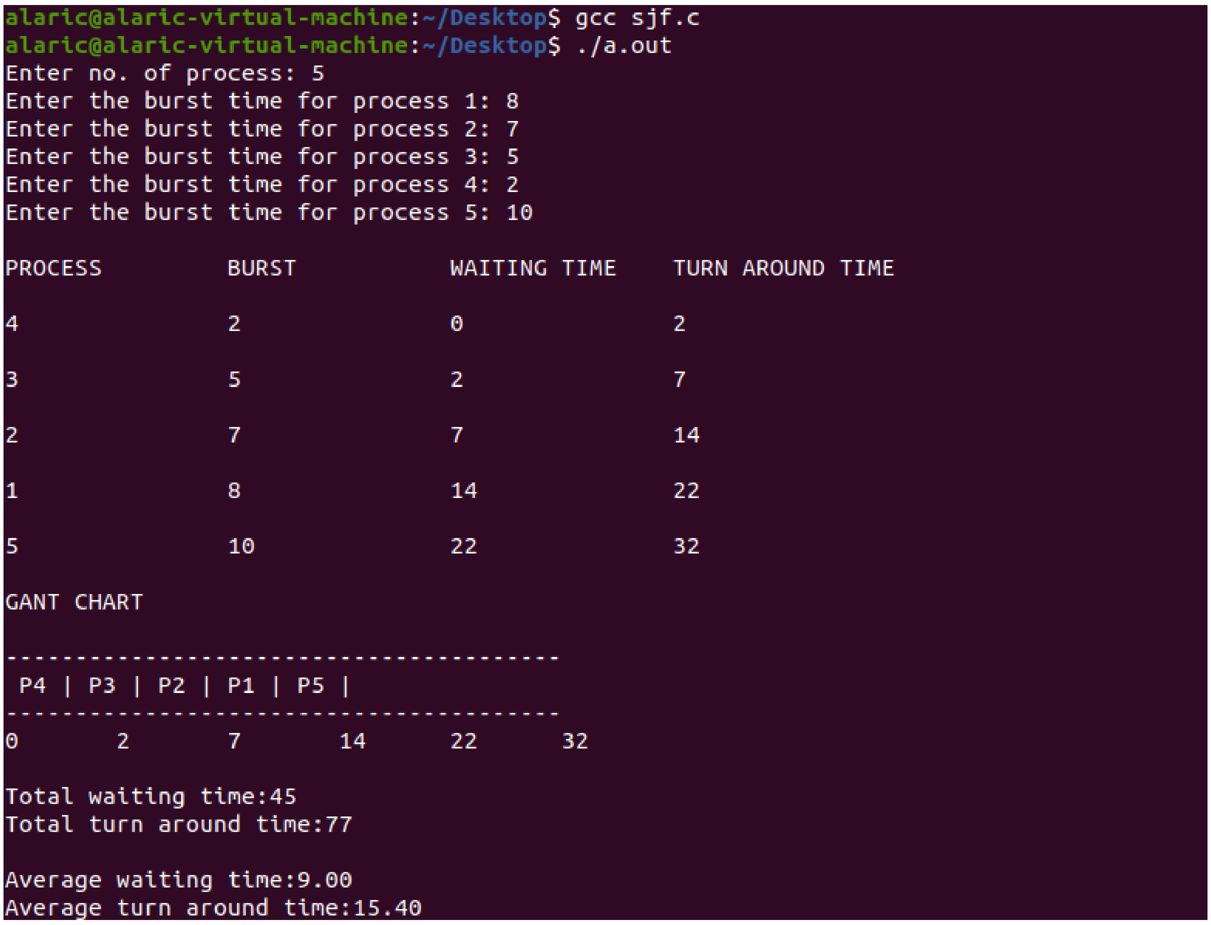


```
avg_wt=(float)total/n;
total=0;
printf("\nPROCESS\t\tBURST\t\tWAITING TIME\tTURN AROUND TIME\n");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\n%d\t\t%d\t\t%d\t\t%d\n",p[i],bt[i],wt[i],tat[i]);
}
printf("\nGANT CHART\n\n");
for(i=0;i<n;i++)
{
printf("--------");
}
printf("\n");
for(i=0;i<n;i++)
{
printf("  P%d  |",p[i]);
}
printf("\n");
for(i=0;i<n;i++)
{
printf("--------");
}
printf("\n");
for(i=0;i<n;i++)
{
printf("%d\t",wt[i]);
}
for(i=4;i<n;i++)
{
printf("%d\t",wt[n-1]+bt[n-1]);
}
for(i=0;i<n;i++)
{
tatime=tatime+tat[i];
wtime=wtime+wt[i];
```



```
}
avg_tat=(float)total/n;
printf("\n\nTotal waiting time:%d",wtime);
printf("\nTotal turn around time:%d",tatime);
printf("\n\nAverage waiting time:%.2f",avg_wt);
printf("\nAverage turn around time:%.2f\n",avg_tat);
}
```

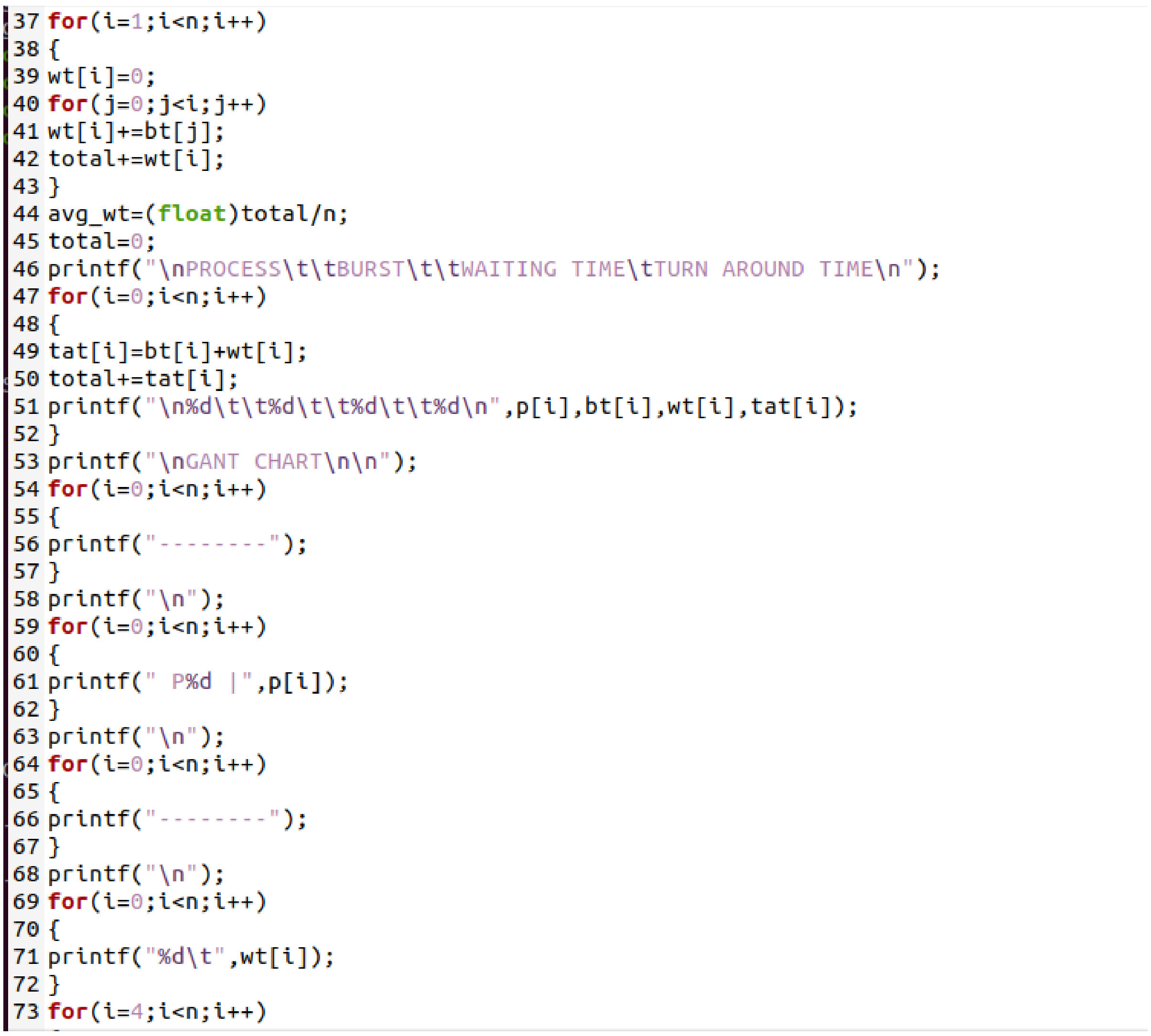
Output:


```
alaric@alaric-virtual-machine:~/Desktop$ gcc sjf.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
Enter no. of process: 5
Enter the burst time for process 1: 8
Enter the burst time for process 2: 7
Enter the burst time for process 3: 5
Enter the burst time for process 4: 2
Enter the burst time for process 5: 10

PROCESS         BURST           WAITING TIME    TURN AROUND TIME

4               2               0               2

3               5               2               7

2               7               7               14

1               8               14              22

5               10              22              32

GANT CHART

----------------------------------------
 P4 | P3 | P2 | P1 | P5 |
----------------------------------------
0       2       7       14      22      32

Total waiting time:45
Total turn around time:77

Average waiting time:9.00
Average turn around time:15.40
```

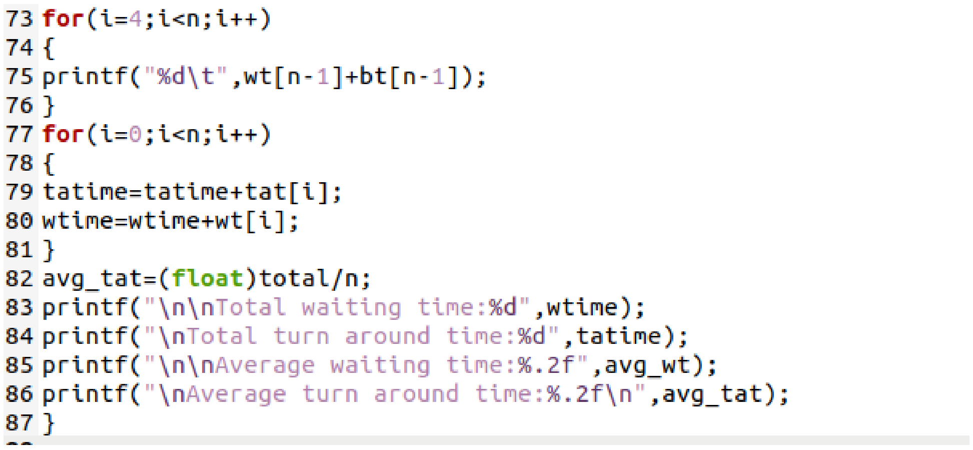
<u>Priority Scheduling:</u>

Code:

```c
#include<stdio.h>
int main()
{
int bt[10],p[10],wt[10],tat[10],pr[10],i,j,n,total=0,pos,temp,wtime=0,tatime=0;
float avg_wt,avg_tat;
printf("Enter no. of process: ");
scanf("%d",&n);
printf("Enter the burst time and priority for processes: ");
for(i=0;i<n;i++)
{
printf("\nP[%d]\n",i+1);
printf("Burst time: ");
scanf("%d",&bt[i]);
printf("Priority: ");
scanf("%d",&pr[i]);
p[i]=i+1;
}
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(pr[j]<pr[pos])
pos=j;
}
temp=pr[i];
pr[i]=pr[pos];
pr[pos]=temp;
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0;
for(i=1;i<n;i++)
```

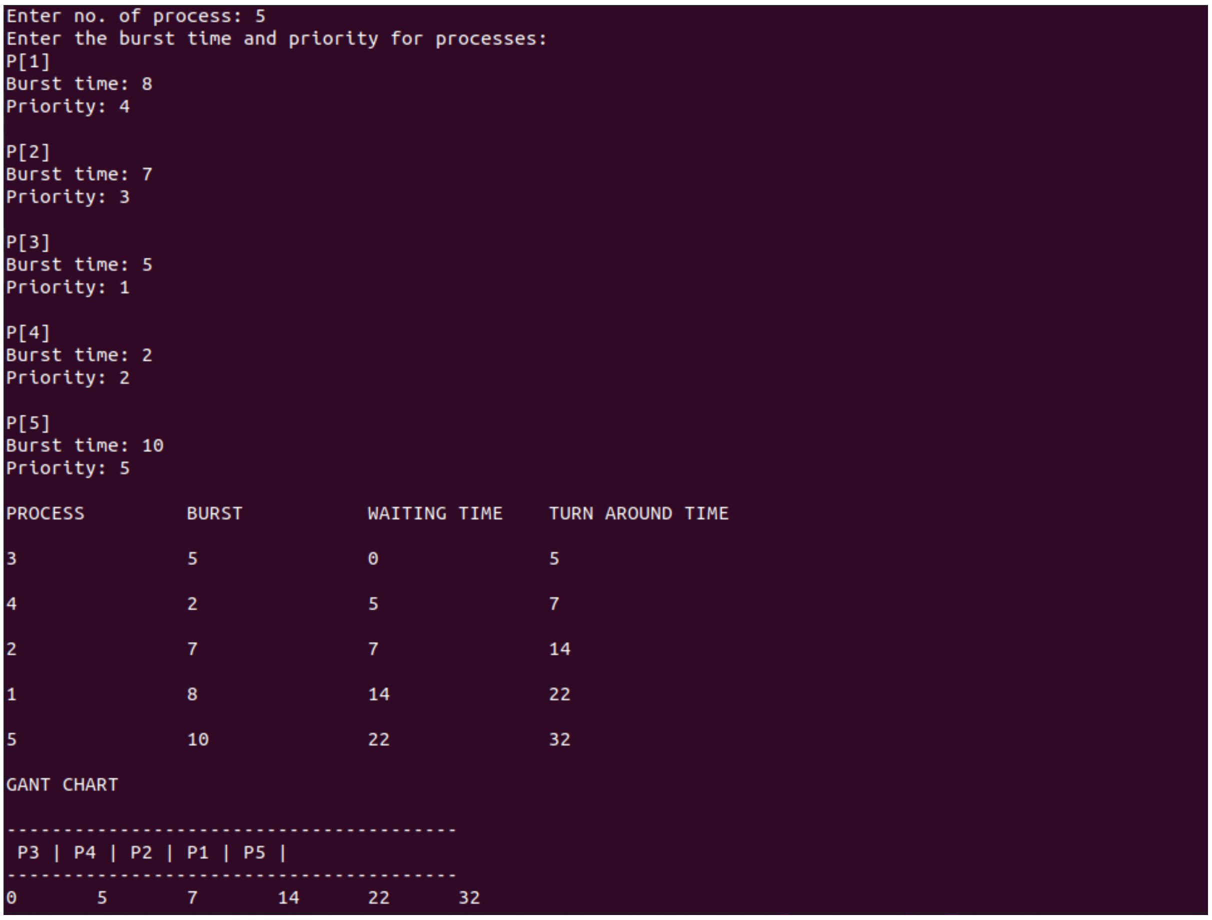


```
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=(float)total/n;
total=0;
printf("\nPROCESS\t\tBURST\t\tWAITING TIME\tTURN AROUND TIME\n");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\n%d\t\t%d\t\t%d\t\t%d\n",p[i],bt[i],wt[i],tat[i]);
}
printf("\nGANT CHART\n\n");
for(i=0;i<n;i++)
{
printf("--------");
}
printf("\n");
for(i=0;i<n;i++)
{
printf(" P%d |",p[i]);
}
printf("\n");
for(i=0;i<n;i++)
{
printf("--------");
}
printf("\n");
for(i=0;i<n;i++)
{
printf("%d\t",wt[i]);
}
for(i=4;i<n;i++)
```

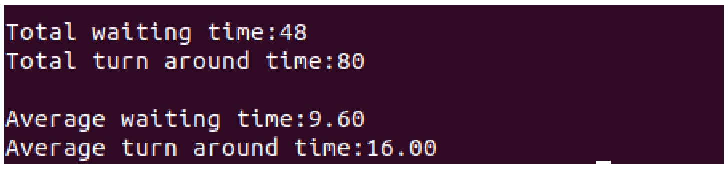


```
for(i=4;i<n;i++)
{
printf("%d\t",wt[n-1]+bt[n-1]);
}
for(i=0;i<n;i++)
{
tatime=tatime+tat[i];
wtime=wtime+wt[i];
}
avg_tat=(float)total/n;
printf("\n\nTotal waiting time:%d",wtime);
printf("\nTotal turn around time:%d",tatime);
printf("\n\nAverage waiting time:%.2f",avg_wt);
printf("\nAverage turn around time:%.2f\n",avg_tat);
}
```

Output:



```
Enter no. of process: 5
Enter the burst time and priority for processes:
P[1]
Burst time: 8
Priority: 4

P[2]
Burst time: 7
Priority: 3

P[3]
Burst time: 5
Priority: 1

P[4]
Burst time: 2
Priority: 2

P[5]
Burst time: 10
Priority: 5

PROCESS         BURST           WAITING TIME    TURN AROUND TIME

3               5               0               5

4               2               5               7

2               7               7               14

1               8               14              22

5               10              22              32

GANT CHART

----------------------------------------
 P3 | P4 | P2 | P1 | P5 |
----------------------------------------
0       5       7       14      22      32
```



```
Total waiting time:48
Total turn around time:80

Average waiting time:9.60
Average turn around time:16.00
```

<u>Round Robin Scheduling:</u>

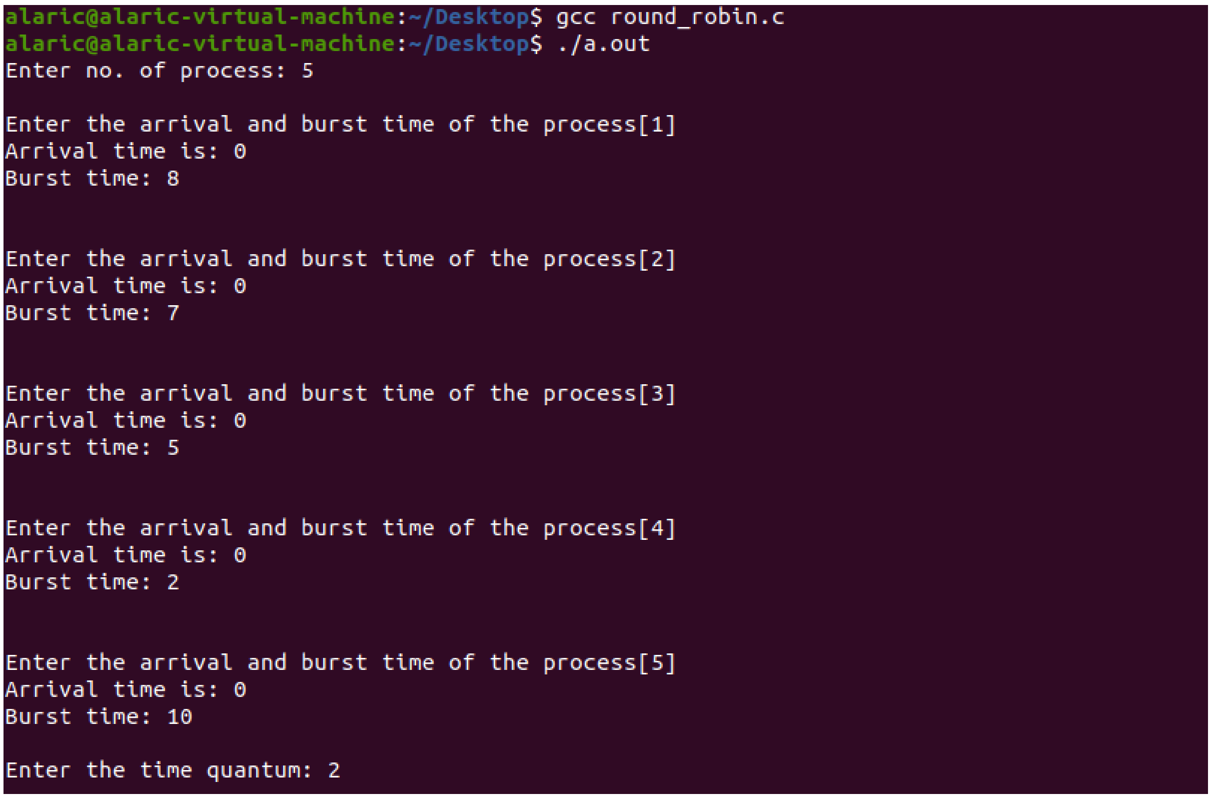
Code:

```c
#include<stdio.h>
int main()
{
int i,j,n,time,remain,flag=0,quantum;
int wt=0,tat=0,at[10],bt[10],rt[10],tatime=0,wtime=0,p[10];
printf("Enter no. of process: ");
scanf("%d",&n);
remain=n;
for(i=0;i<n;i++)
{
printf("\nEnter the arrival and burst time of the process[%d]\n",i+1);
printf("Arrival time is: ");
scanf("%d",&at[i]);
printf("Burst time: ");
scanf("%d",&bt[i]);
rt[i]=bt[i];
printf("\n");
}
printf("Enter the time quantum: ");
scanf("%d",&quantum);
printf("\nPROCESS\t\tBURST\tTURN AROUND TIME\tWAITING TIME\n");
for(time=0,i=0;remain!=0;)
{
if(rt[i]<=quantum&&rt[i]>0)
{
time+=rt[i];
rt[i]=0;
flag=1;
}
else if(rt[i]>0)
{
rt[i]-=quantum;
time+=quantum;
}
if(rt[i]==0&&flag==1)
{
remain--;
```

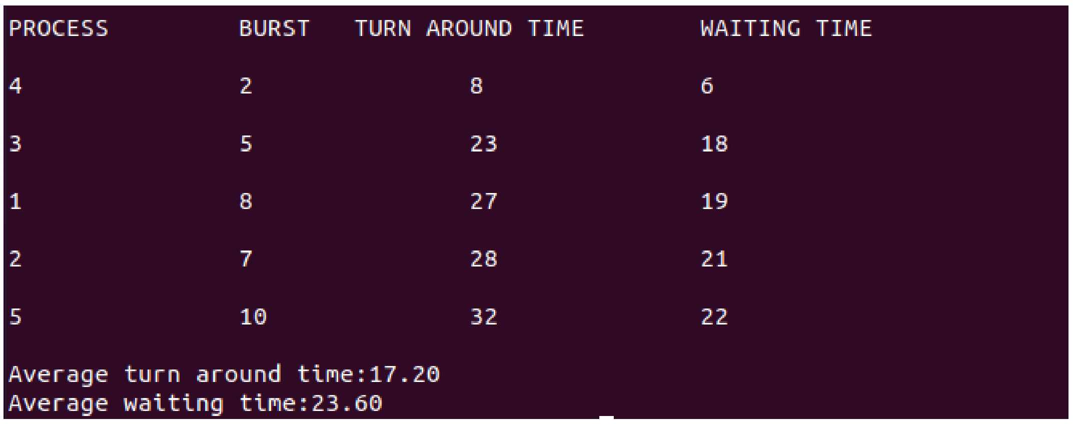
```c
remain--;
```

```c
printf("\n%d\t\t%d\t\t%d\t\t%d\n",i+1,bt[i],time-at[i],time-at[i]-bt[i]);
wt+=time-at[i]-bt[i];
tat=tat+time-at[i];
flag=0;
}
if(i==n-1)
{
i=0;
}
else if(at[i+1]<=time)
{
i++;
}
else
{
i=0;
}
}
printf("\nAverage turn around time:%.2f",wt*1.0/n);
printf("\nAverage waiting time:%.2f",tat*1.0/n);
printf("\n");
return 0;
}
```

Output:



```
alaric@alaric-virtual-machine:~/Desktop$ gcc round_robin.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
Enter no. of process: 5

Enter the arrival and burst time of the process[1]
Arrival time is: 0
Burst time: 8


Enter the arrival and burst time of the process[2]
Arrival time is: 0
Burst time: 7


Enter the arrival and burst time of the process[3]
Arrival time is: 0
Burst time: 5


Enter the arrival and burst time of the process[4]
Arrival time is: 0
Burst time: 2


Enter the arrival and burst time of the process[5]
Arrival time is: 0
Burst time: 10

Enter the time quantum: 2
```



```
PROCESS          BURST    TURN AROUND TIME         WAITING TIME

4                2                8                6

3                5                23               18

1                8                27               19

2                7                28               21

5                10               32               22

Average turn around time:17.20
Average waiting time:23.60
```

Among these scheduling methods SJF has least average waiting time and average turnaround time and hence it is the best scheduling method for ensuring high system performance.