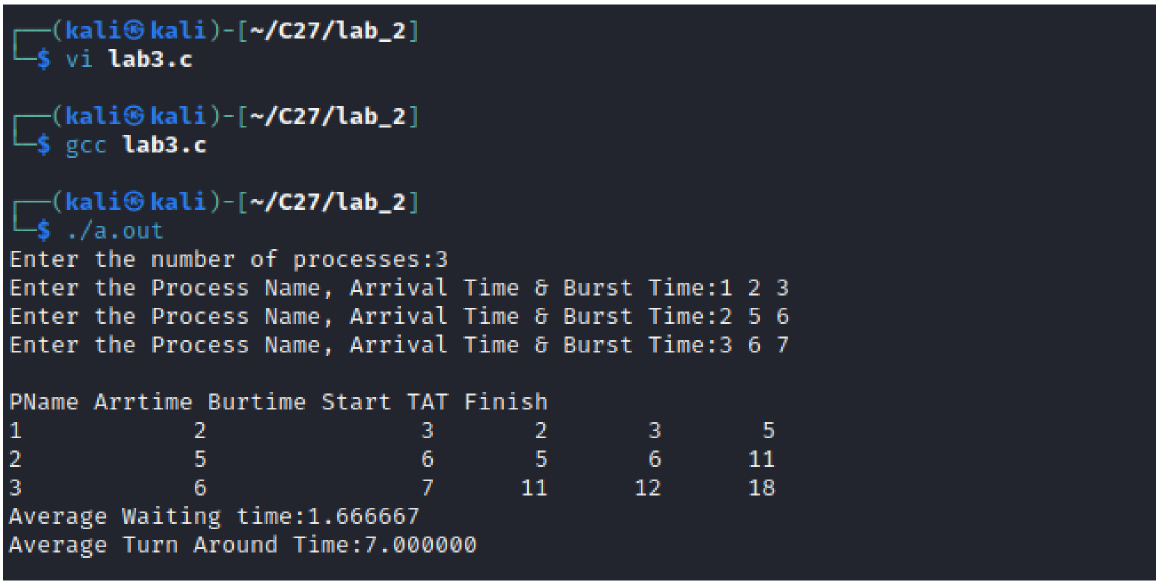
Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering - Cyber Security
B.Tech. 4th Semester , Session: 2023-2024

Student Name:	Aditya.Rajesh.Wanwade
Roll No:	C2-27
Practical No:	3
Aim:	Write C programs to simulate CPU scheduling algorithms: FCFS, SJF, and Round Robin and SRTF

a)To write a C program to implement FCFS CPU scheduling algorithm:

```
#include<stdio.h>
int main()
{
char pn[10][10];
int arr[10],bur[10],star[10],finish[10],tat[10],wt[10],i,n;
int totwt=0,tottat=0;
printf("Enter the number of processes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the Process Name, Arrival Time & Burst Time:");
scanf("%s%d%d",&pn[i],&arr[i],&bur[i]);
}
for(i=0;i<n;i++)
{
if(i==0)
{
star[i]=arr[i];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
else
{
star[i]=finish[i-1];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
}
printf("\nPName Arrtime Burtime Start TAT Finish");
for(i=0;i<n;i++)
{
printf("\n%s\t%6d\t\t%6d\t%6d\t%6d\t%6d",pn[i],arr[i],bur[i],star[i],tat[i],finish[i]);
totwt+=wt[i];
tottat+=tat[i];
}
printf("\nAverage Waiting time:%f", (float)totwt/n);
printf("\nAverage Turn Around Time:%f", (float)tottat/n);
}
```

OUTPUT: (All the test cases are included):

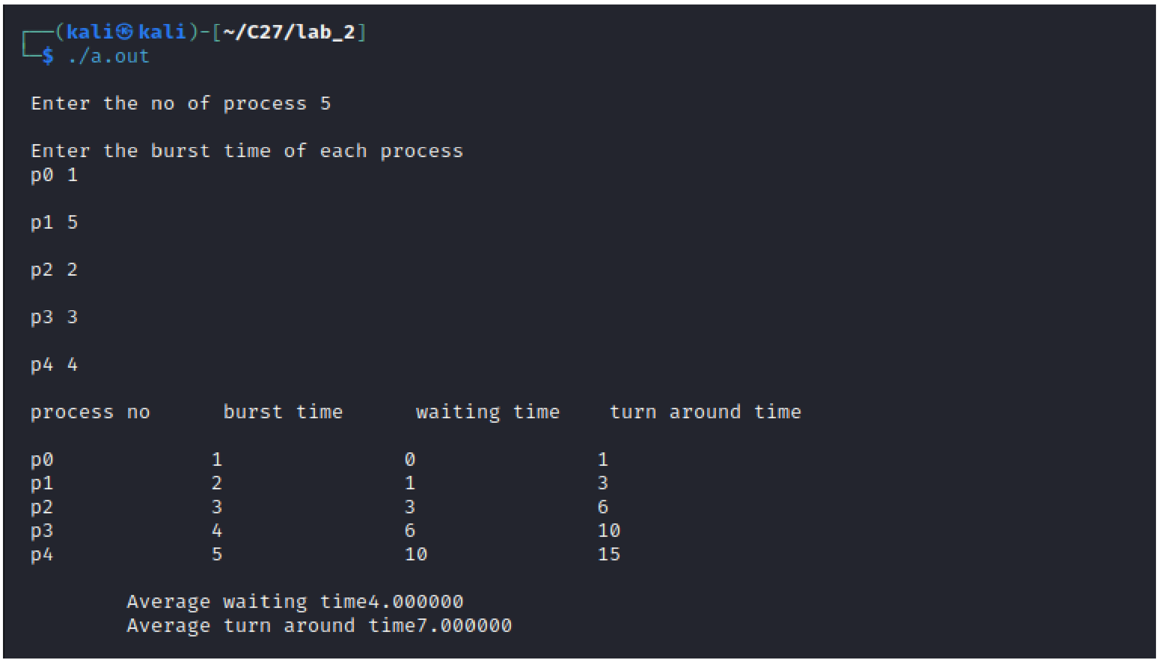


```
┌──(kali㉿kali)-[~/C27/lab_2]
└─$ vi lab3.c

┌──(kali㉿kali)-[~/C27/lab_2]
└─$ gcc lab3.c

┌──(kali㉿kali)-[~/C27/lab_2]
└─$ ./a.out
Enter the number of processes:3
Enter the Process Name, Arrival Time & Burst Time:1 2 3
Enter the Process Name, Arrival Time & Burst Time:2 5 6
Enter the Process Name, Arrival Time & Burst Time:3 6 7

PName Arrtime Burtime Start TAT Finish
1            2               3       2       3       5
2            5               6       5       6      11
3            6               7      11      12      18
Average Waiting time:1.666667
Average Turn Around Time:7.000000
```

b) To write a C program to implement SJF CPU scheduling algorithm:

```
#include<stdio.h>
#include<string.h>
int main()
{
int i=0,pno[10],bt[10],n,wt[10],temp=0,j,tt[10];
float sum,at;
printf("\n Enter the no of process ");
scanf("\n %d",&n);
printf("\n Enter the burst time of each process");
for(i=0;i<n;i++)
{
printf("\n p%d",i);
scanf("%d",&bt[i]);
}
for(i=0;i<n-1;i++)
{
for(j=i+1;j<n;j++)
{
if(bt[i]>bt[j])
{
temp=bt[i];
bt[i]=bt[j];
bt[j]=temp;
temp=pno[i];
pno[i]=pno[j];
pno[j]=temp;
}
}
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=bt[i-1]+wt[i-1];
sum=sum+wt[i];
}
printf("\n process no \t burst time\t waiting time \t turn around time\n");
for(i=0;i<n;i++)
{
tt[i]=bt[i]+wt[i];
at+=tt[i];
printf("\n p%d\t\t%d\t\t%d\t\t%d",i,bt[i],wt[i],tt[i]);
}
printf("\n\n\t Average waiting time%f\n\t Average turn around time%f", sum/n, at/n);
}
```

<u>OUTPUT: </u>(All the test cases are included):

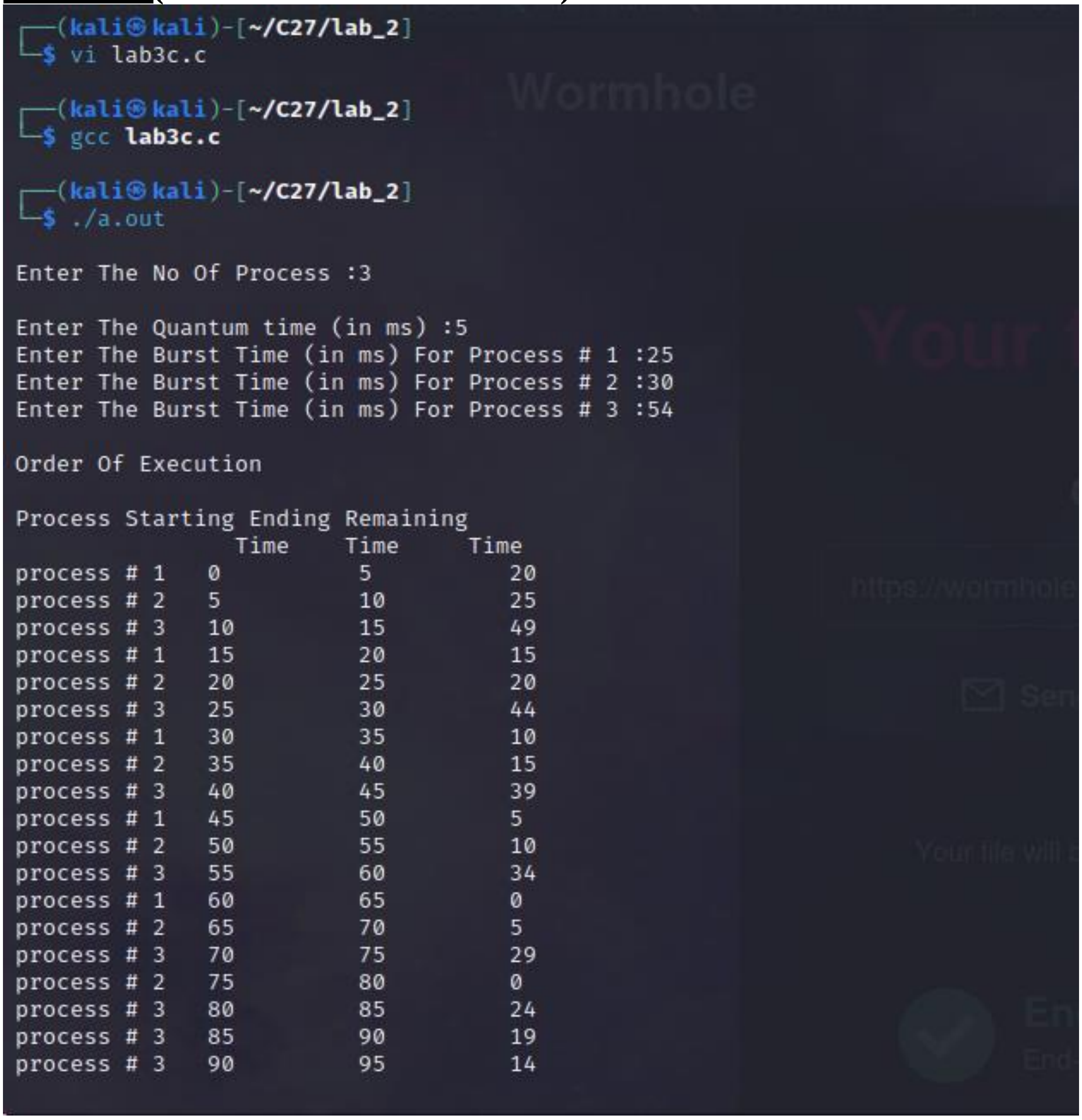


```
┌──(kali㉿kali)-[~/C27/lab_2]
└─$ ./a.out

 Enter the no of process 5

 Enter the burst time of each process
 p0 1

 p1 5

 p2 2

 p3 3

 p4 4

 process no      burst time      waiting time    turn around time

 p0             1               0               1
 p1             2               1               3
 p2             3               3               6
 p3             4               6               10
 p4             5               10              15

         Average waiting time4.000000
         Average turn around time7.000000
```

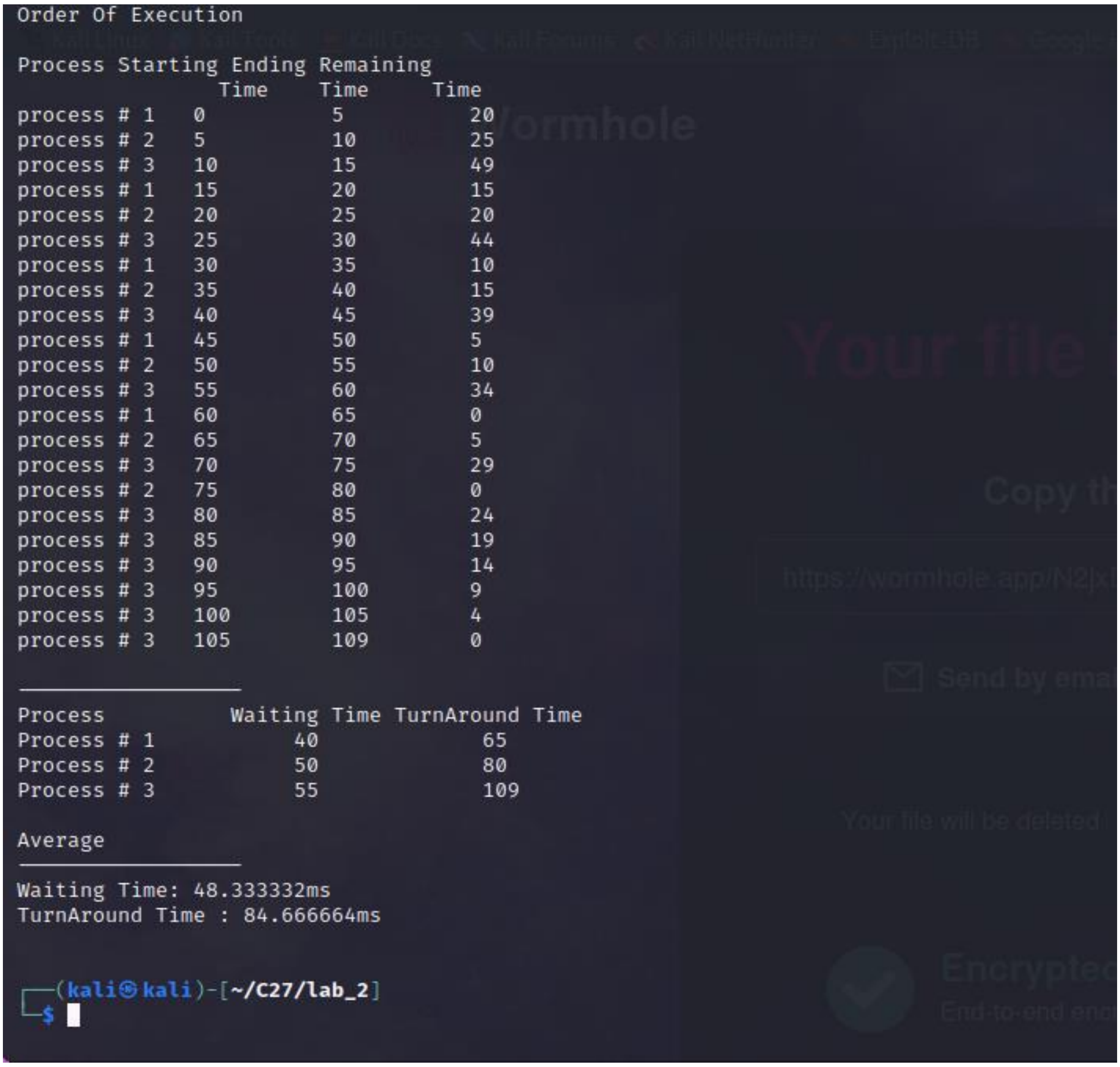
c) To write a C program to implement Round Robin CPU scheduling algorithm.

```
#include<stdio.h>
struct process
{
int burst,wait,comp,f;
}p[20]={0,0};
int main()
{
int n,i,j,totalwait=0,totalturn=0,quantum,flag=1,time=0;
printf("\nEnter The No Of Process :");
scanf("%d",&n);
printf("\nEnter The Quantum time (in ms) :");
scanf("%d",&quantum);
for(i=0;i<n;i++)
{
printf("Enter The Burst Time (in ms) For Process #%2d :",i+1);
scanf("%d",&p[i].burst);
p[i].f=1;
}
printf("\nOrder Of Execution \n");
printf("\nProcess Starting Ending Remaining");
printf("\n\t\tTime \tTime \t Time");
while(flag==1)
{
flag=0;
for(i=0;i<n;i++)
{
if(p[i].f==1)
{
flag=1;
j=quantum;
if((p[i].burst-p[i].comp)>quantum)
{
p[i].comp+=quantum;
}
else
{
p[i].wait=time-p[i].comp;
j=p[i].burst-p[i].comp;
p[i].comp=p[i].burst;
p[i].f=0;
}
printf("\nprocess # %-3d %-10d %-10d %-10d", i+1, time, time+j,
p[i].burst-p[i].comp);
time+=j;
}
}
}
printf("\n\n------------------");
printf("\nProcess \t Waiting Time TurnAround Time ");
for(i=0;i<n;i++)
{
printf("\nProcess # %-12d%-15d%-15d",i+1,p[i].wait,p[i].wait+p[i].burst);
totalwait=totalwait+p[i].wait;
totalturn=totalturn+p[i].wait+p[i].burst;
}
printf("\n\nAverage\n------------------ ");
printf("\nWaiting Time: %fms",totalwait/(float)n);
printf("\nTurnAround Time : %fms\n\n",totalturn/(float)n);
return 0;
}
```

<u>OUTPUT:</u> (All the test cases are included):



```
┌──(kali㉿kali)-[~/C27/lab_2]
└─$ vi lab3c.c

┌──(kali㉿kali)-[~/C27/lab_2]
└─$ gcc lab3c.c

┌──(kali㉿kali)-[~/C27/lab_2]
└─$ ./a.out

Enter The No Of Process :3

Enter The Quantum time (in ms) :5
Enter The Burst Time (in ms) For Process # 1 :25
Enter The Burst Time (in ms) For Process # 2 :30
Enter The Burst Time (in ms) For Process # 3 :54

Order Of Execution

Process Starting Ending Remaining
                Time    Time      Time
process # 1    0        5          20
process # 2    5        10         25
process # 3    10       15         49
process # 1    15       20         15
process # 2    20       25         20
process # 3    25       30         44
process # 1    30       35         10
process # 2    35       40         15
process # 3    40       45         39
process # 1    45       50         5
process # 2    50       55         10
process # 3    55       60         34
process # 1    60       65         0
process # 2    65       70         5
process # 3    70       75         29
process # 2    75       80         0
process # 3    80       85         24
process # 3    85       90         19
process # 3    90       95         14
```

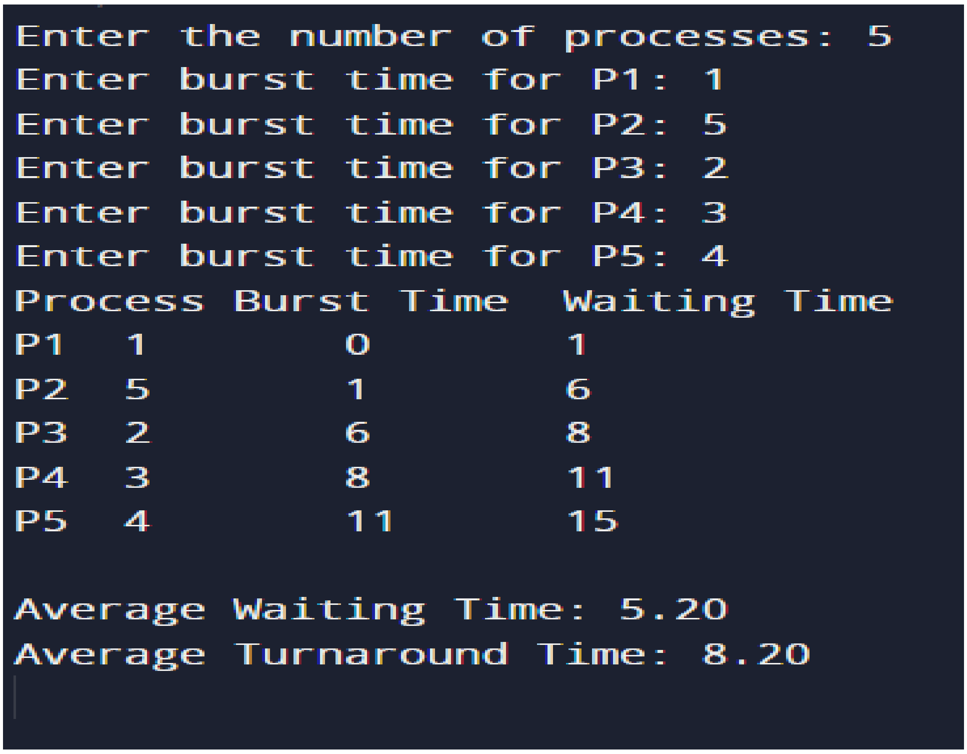


```
Order Of Execution

Process Starting Ending Remaining
                Time     Time      Time
process # 1   0         5         20
process # 2   5         10        25
process # 3   10        15        49
process # 1   15        20        15
process # 2   20        25        20
process # 3   25        30        44
process # 1   30        35        10
process # 2   35        40        15
process # 3   40        45        39
process # 1   45        50        5
process # 2   50        55        10
process # 3   55        60        34
process # 1   60        65        0
process # 2   65        70        5
process # 3   70        75        29
process # 2   75        80        0
process # 3   80        85        24
process # 3   85        90        19
process # 3   90        95        14
process # 3   95        100       9
process # 3   100       105       4
process # 3   105       109       0

__________________

Process           Waiting Time TurnAround Time
Process # 1           40             65
Process # 2           50             80
Process # 3           55             109

Average
__________________

Waiting Time: 48.333332ms
TurnAround Time : 84.666664ms


┌──(kali㉿kali)-[~/C27/lab_2]
└─$ █
```

d)To write a C program to implement SRTF CPU scheduling algorithm:

```
#include <stdio.h>

struct Process {
    int pid; // Process ID
    int burst_time;
};

void calculateTimes(struct Process processes[], int n) {
    int waiting_time[n], turnaround_time[n];
    int total_waiting_time = 0, total_turnaround_time = 0;
    waiting_time[0] = 0;
    for (int i = 1; i < n; i++) {
        waiting_time[i] = waiting_time[i - 1] + processes[i - 1].burst_time;
        total_waiting_time += waiting_time[i];
    }

    for (int i = 0; i < n; i++) {
        turnaround_time[i] = waiting_time[i] + processes[i].burst_time;
        total_turnaround_time += turnaround_time[i];
    }

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burst_time,
               waiting_time[i], turnaround_time[i]);
    }

    double avg_waiting_time = (double)total_waiting_time / n;
    double avg_turnaround_time = (double)total_turnaround_time / n;
    printf("\nAverage Waiting Time: %.2lf\n", avg_waiting_time);
    printf("Average Turnaround Time: %.2lf\n", avg_turnaround_time);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {
        processes[i].pid = i + 1;
        printf("Enter burst time for P%d: ", i + 1);
        scanf("%d", &processes[i].burst_time);
    }
    calculateTimes(processes, n);
    return 0;
}
```

<u>OUTPUT:</u> (All the test cases are included):



```
Enter the number of processes: 5
Enter burst time for P1: 1
Enter burst time for P2: 5
Enter burst time for P3: 2
Enter burst time for P4: 3
Enter burst time for P5: 4
Process Burst Time  Waiting Time
P1   1        0        1
P2   5        1        6
P3   2        6        8
P4   3        8        11
P5   4        11       15

Average Waiting Time: 5.20
Average Turnaround Time: 8.20
```

<u>Result</u>: Thus the program was executed and verified successfully.