Lab 5:
Scheduling Algorithms in OS

Programme	:	**BTech. CSE Core**	Semester	:	**Win 2021-22**
Course	:	**Operating Systems**	Code	:	**CSE2005**
Faculty	:	**Dr. Shyamala L**	Slot	:	**L25+L26**
Name	:	**Hariket Sukesh Kumar Sheth**	Register No.	:	**20BCE1975**

LAB 5

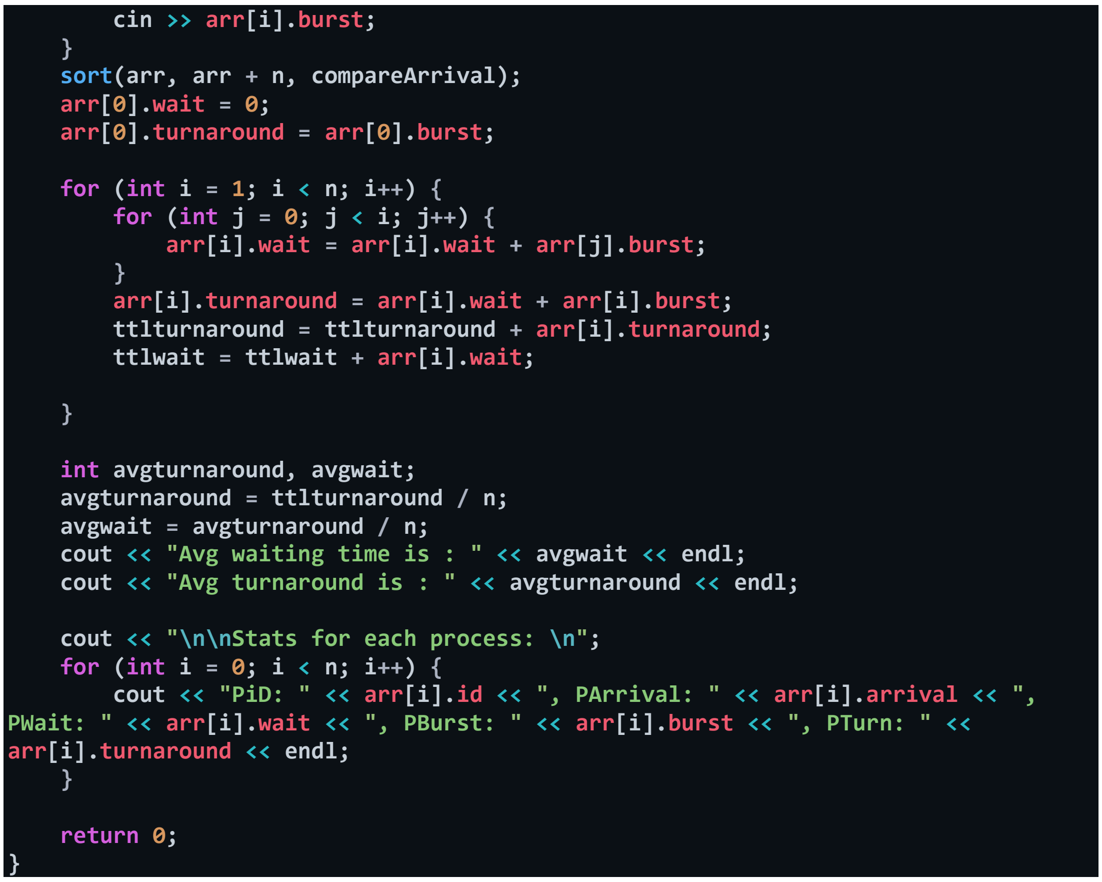
Consider the ready queue of OS, the process are present and maintained with their arrival time and expected burst time for execution. Some processes have priority which is also given. Consider the required data to run different scheduling algorithms and analyse the result with respect to average waiting time and turnaround time.

- In priority, lowest number has highest priority
- For time quantum, t = 2 ms.

S.no	Process ID	Arrival time	Expected Burst time	Priority
1	P1	0	15	3
2	P2	2	2	5
3	P3	4	5	2
4	P4	7	1	1
5	P5	3	7	4

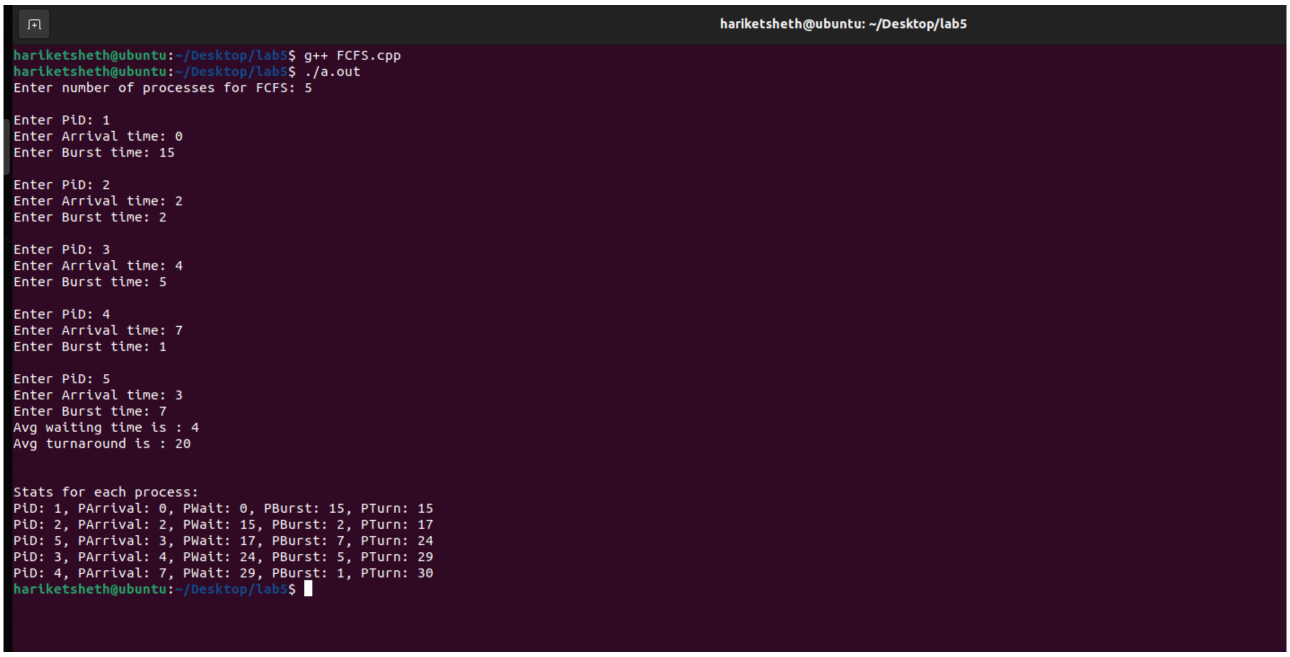
1. Write one single C /C++ program to simulate different scheduling algorithm in OS (functions)
    a. First Come First Serve

```
#include<iostream>
#include<algorithm>
using namespace std;

struct process {
    int id = 0;
    int wait = 0;
    int burst = 0;
    int turnaround = 0;
    int arrival = 0;
};

bool compareArrival(process p1, process p2) {
    return (p1.arrival < p2.arrival);
}

int main() {
    cout << "Enter number of processes for FCFS: ";
    int n;
    int ttlwait = 0, ttlturnaround = 0;
    cin >> n;
    process arr[n];
    for (int i = 0; i < n; i++) {
        cout << "\nEnter PiD: ";
        cin >> arr[i].id;
        cout << "\nEnter Arrival time: ";
        cin >> arr[i].arrival;
        cout << "\nEnter Burst time: ";
```



```
        cin >> arr[i].burst;
    }
    sort(arr, arr + n, compareArrival);
    arr[0].wait = 0;
    arr[0].turnaround = arr[0].burst;

    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            arr[i].wait = arr[i].wait + arr[j].burst;
        }
        arr[i].turnaround = arr[i].wait + arr[i].burst;
        ttlturnaround = ttlturnaround + arr[i].turnaround;
        ttlwait = ttlwait + arr[i].wait;

    }

    int avgturnaround, avgwait;
    avgturnaround = ttlturnaround / n;
    avgwait = avgturnaround / n;
    cout << "Avg waiting time is : " << avgwait << endl;
    cout << "Avg turnaround is : " << avgturnaround << endl;

    cout << "\n\nStats for each process: \n";
    for (int i = 0; i < n; i++) {
        cout << "PiD: " << arr[i].id << ", PArrival: " << arr[i].arrival << ",
PWait: " << arr[i].wait << ", PBurst: " << arr[i].burst << ", PTurn: " <<
arr[i].turnaround << endl;
    }

    return 0;
}
```

OUTPUT:



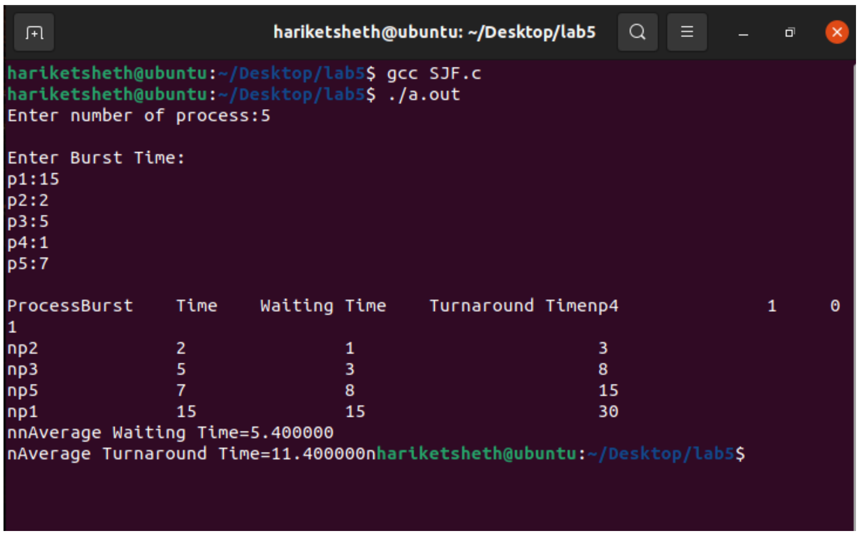
```
hariketsheth@ubuntu:~/Desktop/lab5$ g++ FCFS.cpp
hariketsheth@ubuntu:~/Desktop/lab5$ ./a.out
Enter number of processes for FCFS: 5

Enter PiD: 1
Enter Arrival time: 0
Enter Burst time: 15

Enter PiD: 2
Enter Arrival time: 2
Enter Burst time: 2

Enter PiD: 3
Enter Arrival time: 4
Enter Burst time: 5

Enter PiD: 4
Enter Arrival time: 7
Enter Burst time: 1

Enter PiD: 5
Enter Arrival time: 3
Enter Burst time: 7
Avg waiting time is : 4
Avg turnaround is : 20


Stats for each process:
PiD: 1, PArrival: 0, PWait: 0, PBurst: 15, PTurn: 15
PiD: 2, PArrival: 2, PWait: 15, PBurst: 2, PTurn: 17
PiD: 5, PArrival: 3, PWait: 17, PBurst: 7, PTurn: 24
PiD: 3, PArrival: 4, PWait: 24, PBurst: 5, PTurn: 29
PiD: 4, PArrival: 7, PWait: 29, PBurst: 1, PTurn: 30
hariketsheth@ubuntu:~/Desktop/lab5$
```

Average Waiting Time = 4
Average Turnaround Time = 20

b. Shortest Job First

```c
#include<stdio.h>
 int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
        float avg_wt,avg_tat;
        printf("Enter number of process:");
        scanf("%d",&n);
        printf("\nEnter Burst Time:\n");
        for(i=0;i<n;i++)    {
            printf("p%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;
        total=0;
    printf("\nProcessBurst\tTime\tWaiting Time\tTurnaround Time");
        for(i=0;i<n;i++)        {
            tat[i]=bt[i]+wt[i];
            total+=tat[i];
            printf("np%d\t\t%d\t\t%d\t\t\t%d\n",p[i],bt[i],wt[i],tat[i]);
        }
        avg_tat=(float)total/n;
        printf("nnAverage Waiting Time=%f\n",avg_wt);
        printf("nAverage Turnaround Time=%fn",avg_tat);
}
```

OUTPUT:

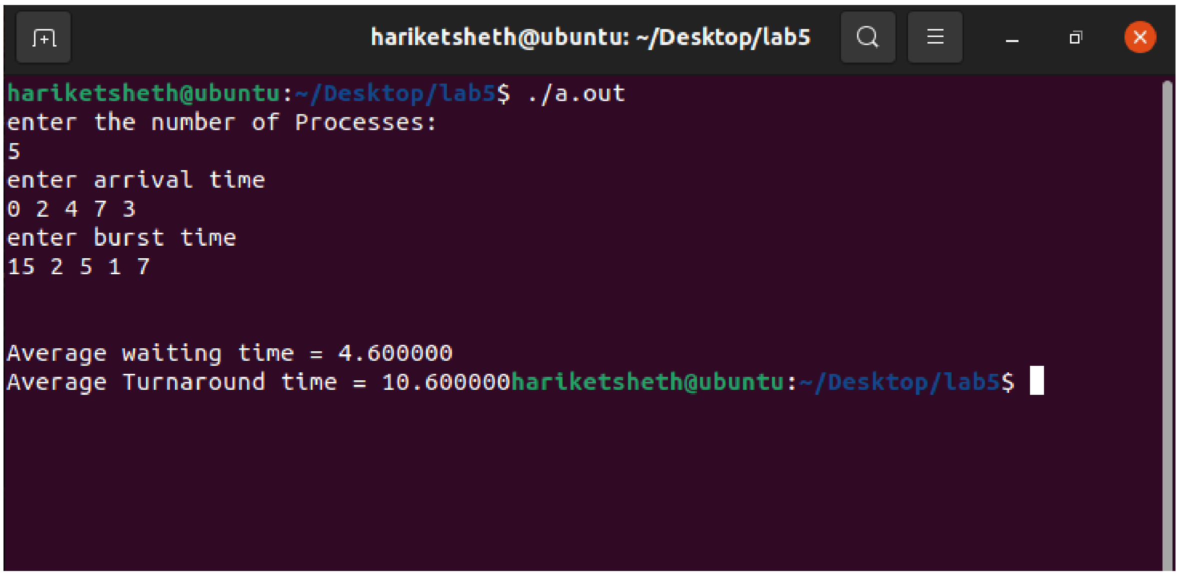


```
hariketsheth@ubuntu:~/Desktop/lab5$ gcc SJF.c
hariketsheth@ubuntu:~/Desktop/lab5$ ./a.out
Enter number of process:5

Enter Burst Time:
p1:15
p2:2
p3:5
p4:1
p5:7

ProcessBurst    Time    Waiting Time    Turnaround Timenp4              1       0
1
np2             2               1                       3
np3             5               3                       8
np5             7               8                       15
np1             15              15                      30
nnAverage Waiting Time=5.400000
nAverage Turnaround Time=11.400000nhariketsheth@ubuntu:~/Desktop/lab5$
```

c. Shortest Remaining Time First

```c
#include <stdio.h>
int main()
{
    int a[10],b[10],x[10],i,smallest,count=0,time,n;
    double avg=0,tt=0,end;
    printf("enter the number of Processes:\n");
    scanf("%d",&n);
    printf("enter arrival time\n");
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    printf("enter burst time\n");
    for(i=0;i<n;i++)
    scanf("%d",&b[i]);
    for(i=0;i<n;i++)
    x[i]=b[i];

    b[9]=9999;

    for(time=0;count!=n;time++)
    {
        smallest=9;
        for(i=0;i<n;i++)
        {
            if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )
            smallest=i;
        }
        b[smallest]--;
        if(b[smallest]==0)
        {
            count++;
            end=time+1;
            avg=avg+end-a[smallest]-x[smallest];
            tt= tt+end-a[smallest];
        }
    }
    printf("\n\nAverage waiting time = %lf\n",avg/n);
        printf("Average Turnaround time = %lf",tt/n);
        return 0;
}
```

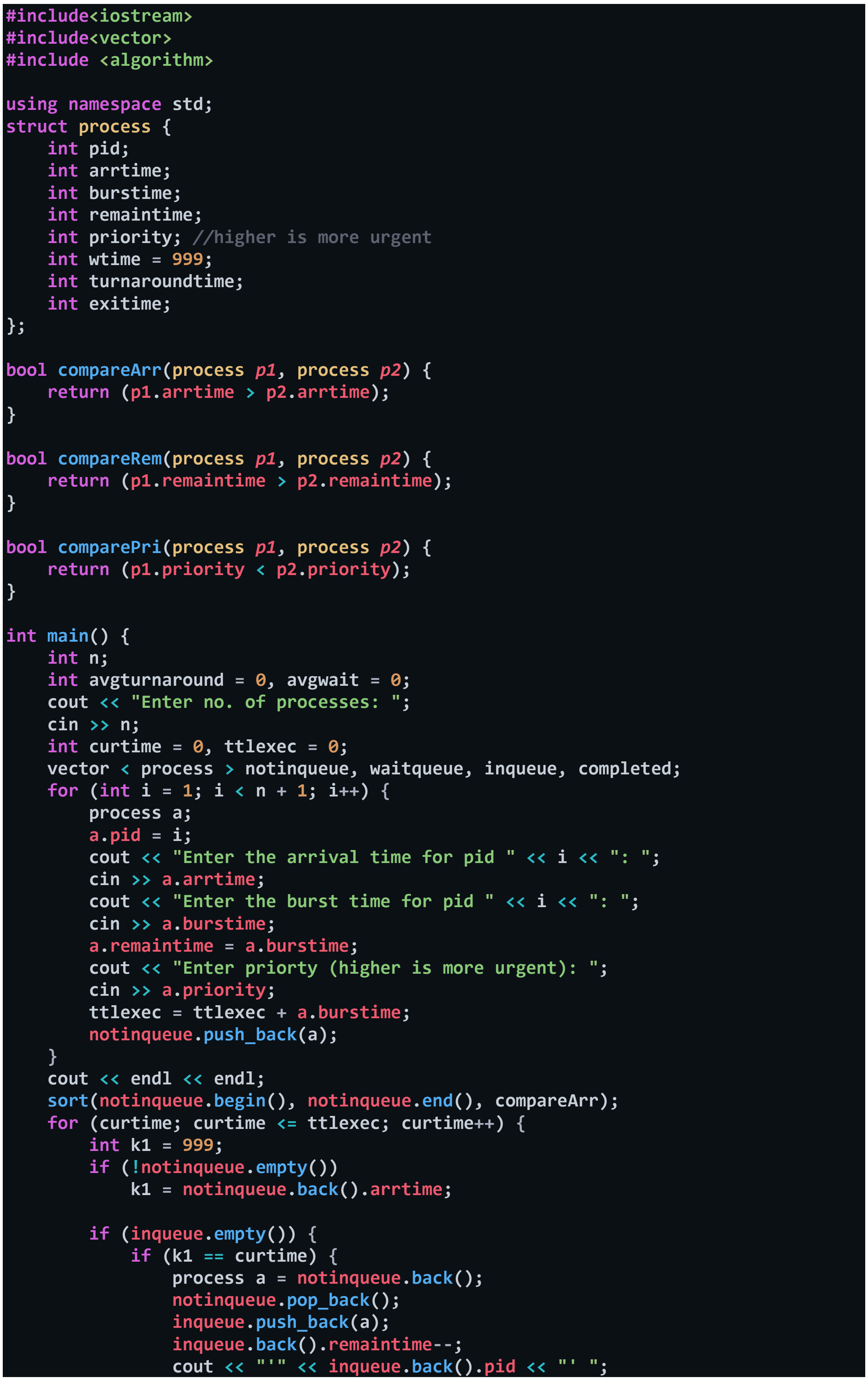
OUTPUT:



```
hariketsheth@ubuntu:~/Desktop/lab5$ ./a.out
enter the number of Processes:
5
enter arrival time
0 2 4 7 3
enter burst time
15 2 5 1 7


Average waiting time = 4.600000
Average Turnaround time = 10.600000hariketsheth@ubuntu:~/Desktop/lab5$
```

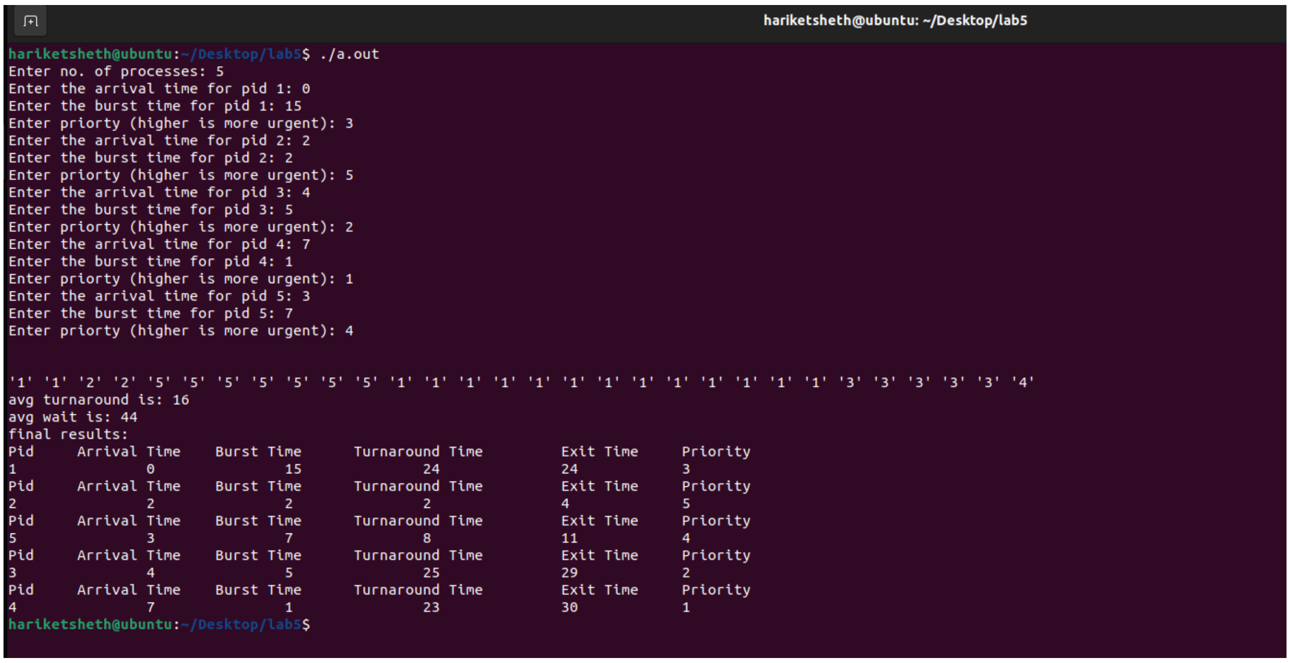
d. Priority

i. Non-Pre-emptive



```cpp
#include<iostream>
#include<vector>
#include <algorithm>

using namespace std;
struct process {
    int pid;
    int arrtime;
    int burstime;
    int remaintime;
    int priority; //higher is more urgent
    int wtime = 999;
    int turnaroundtime;
    int exitime;
};

bool compareArr(process p1, process p2) {
    return (p1.arrtime > p2.arrtime);
}

bool compareRem(process p1, process p2) {
    return (p1.remaintime > p2.remaintime);
}

bool comparePri(process p1, process p2) {
    return (p1.priority < p2.priority);
}

int main() {
    int n;
    int avgturnaround = 0, avgwait = 0;
    cout << "Enter no. of processes: ";
    cin >> n;
    int curtime = 0, ttlexec = 0;
    vector < process > notinqueue, waitqueue, inqueue, completed;
    for (int i = 1; i < n + 1; i++) {
        process a;
        a.pid = i;
        cout << "Enter the arrival time for pid " << i << ": ";
        cin >> a.arrtime;
        cout << "Enter the burst time for pid " << i << ": ";
        cin >> a.burstime;
        a.remaintime = a.burstime;
        cout << "Enter priorty (higher is more urgent): ";
        cin >> a.priority;
        ttlexec = ttlexec + a.burstime;
        notinqueue.push_back(a);
    }
    cout << endl << endl;
    sort(notinqueue.begin(), notinqueue.end(), compareArr);
    for (curtime; curtime <= ttlexec; curtime++) {
        int k1 = 999;
        if (!notinqueue.empty())
            k1 = notinqueue.back().arrtime;

        if (inqueue.empty()) {
            if (k1 == curtime) {
                process a = notinqueue.back();
                notinqueue.pop_back();
                inqueue.push_back(a);
                inqueue.back().remaintime--;
                cout << "'" << inqueue.back().pid << "' ";
                inqueue.back().wtime = 0;
                avgwait = avgwait + curtime;
                if (inqueue.back().remaintime == 0) {
                    inqueue.back().exitime = curtime + 1;
                    inqueue.back().turnaroundtime = inqueue.back().exitime -
inqueue.back().arrtime;
                    avgturnaround = avgturnaround + inqueue.back().turnaroundtime;
                    completed.push_back(inqueue.back());
                    inqueue.pop_back();
                }
            }
        } else {
            if (k1 == curtime) {
                process a = notinqueue.back();
                notinqueue.pop_back();
                waitqueue.push_back(a);
            }
            if (!inqueue.empty()) {
                inqueue.back().remaintime--;
                cout << "'" << inqueue.back().pid << "' ";
            }
            if (inqueue.back().wtime != 999) {
                inqueue.back().wtime = curtime;
                avgwait = avgwait + curtime;
            }
            if (inqueue.back().remaintime == 0) {
                inqueue.back().exitime = curtime + 1;
                inqueue.back().turnaroundtime = inqueue.back().exitime -
inqueue.back().arrtime;
                avgturnaround = avgturnaround + inqueue.back().turnaroundtime;
                process a = inqueue.back();
                completed.push_back(a);
                inqueue.pop_back();
                sort(waitqueue.begin(), waitqueue.end(), comparePri); //for non-
primitive
                inqueue.push_back(waitqueue.back());
                waitqueue.pop_back();
            }
        }
    }

    avgturnaround = avgturnaround / n;
    avgwait = avgwait / n;
    cout << "\navg turnaround is: " << avgturnaround;
    cout << "\navg wait is: " << avgwait << endl;
    cout << "final results: " << endl;
    sort(completed.begin(), completed.end(), compareArr);
    for (int i = 0; i < n; i++) {
        cout << "Pid\tArrival Time\tBurst Time\tTurnaround Time\t\tExit
Time\tPriority\n";
        cout << completed.back().pid << "\t\t" << completed.back().arrtime <<
"\t\t" <<
            completed.back().burstime << "\t\t" << completed.back().turnaroundtime
<< "\t\t" <<
            completed.back().exitime << "\t\t" << completed.back().priority <<
"\n";
        completed.pop_back();
    }
    return 0;
}
```

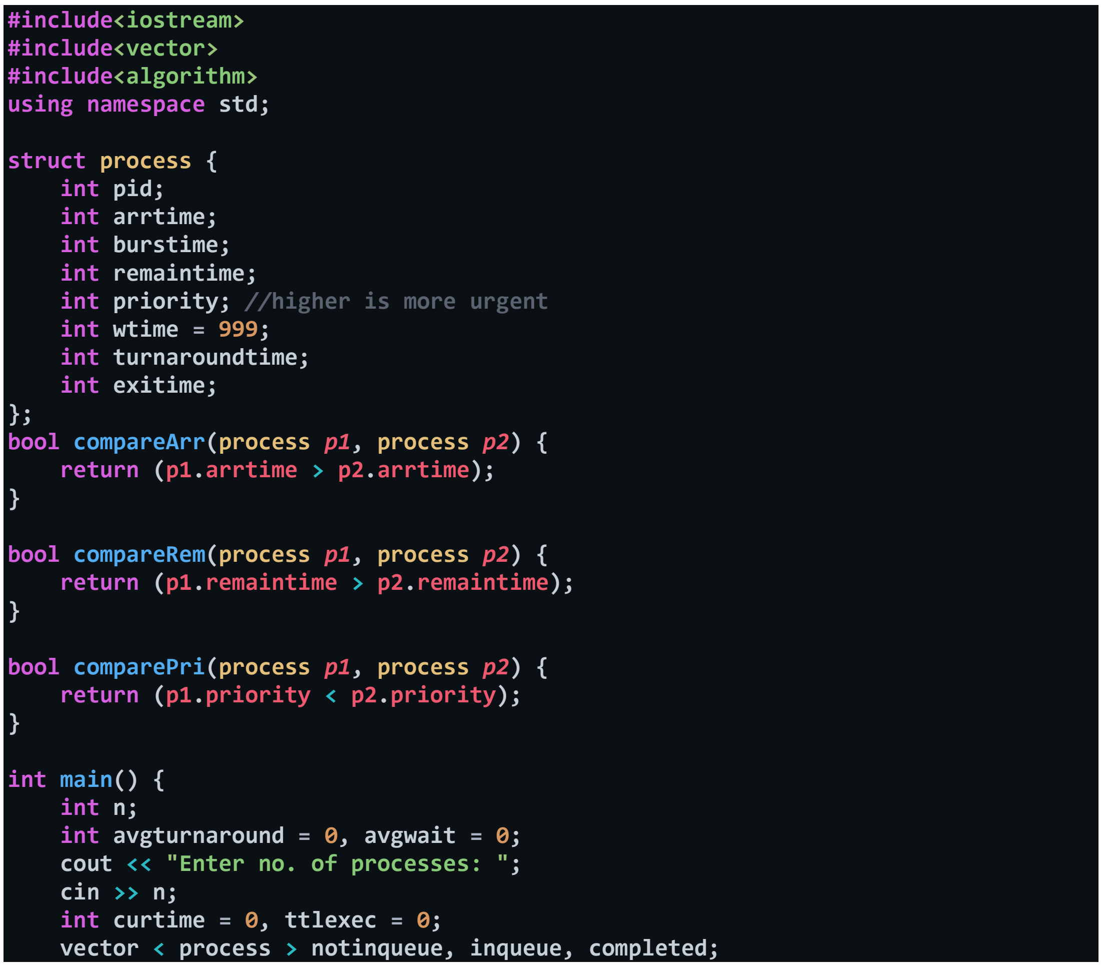
OUTPUT:



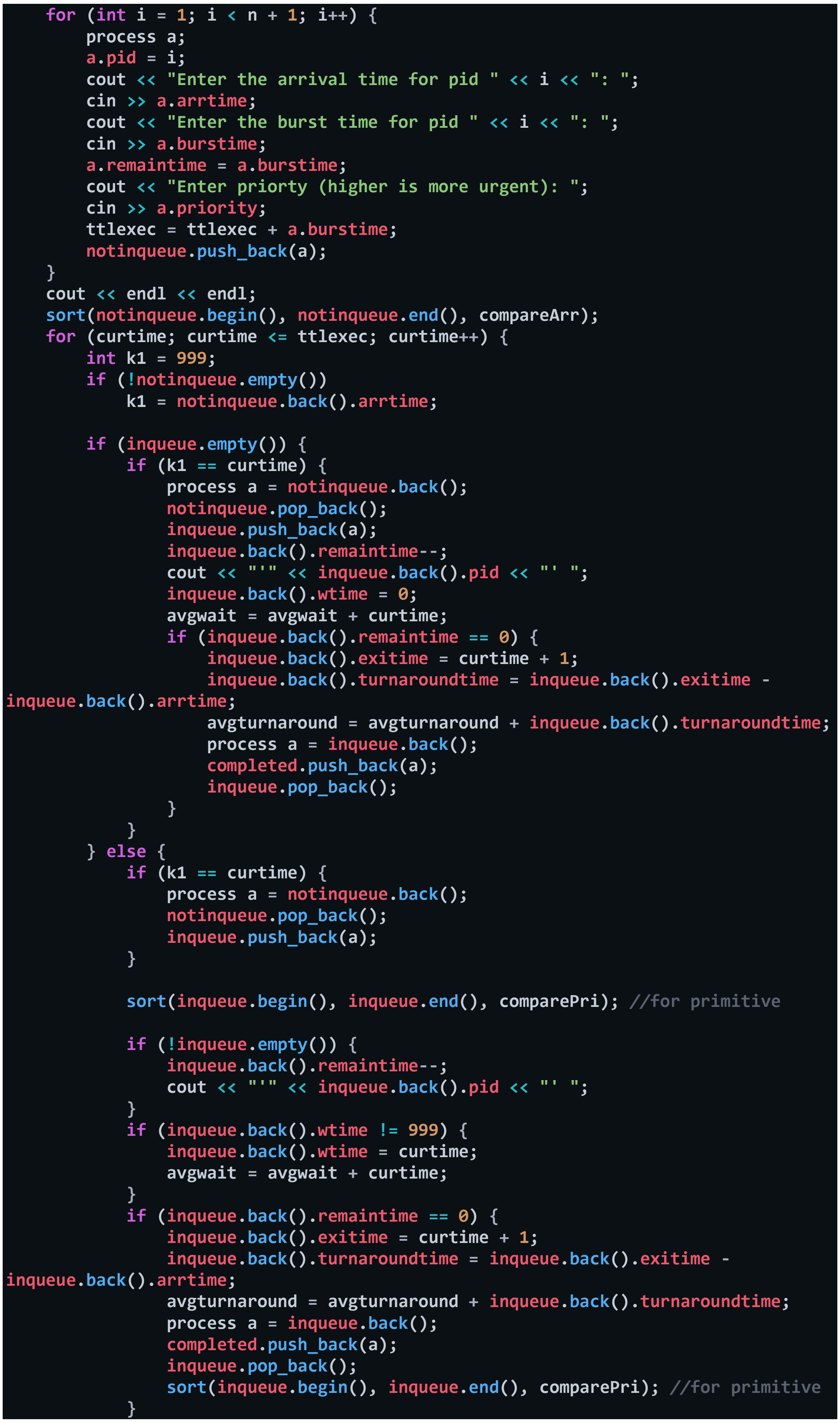
hariketsheth@ubuntu: ~/Desktop/lab5

```
hariketsheth@ubuntu:~/Desktop/lab5$ ./a.out
Enter no. of processes: 5
Enter the arrival time for pid 1: 0
Enter the burst time for pid 1: 15
Enter priorty (higher is more urgent): 3
Enter the arrival time for pid 2: 2
Enter the burst time for pid 2: 2
Enter priorty (higher is more urgent): 5
Enter the arrival time for pid 3: 4
Enter the burst time for pid 3: 5
Enter priorty (higher is more urgent): 2
Enter the arrival time for pid 4: 7
Enter the burst time for pid 4: 1
Enter priorty (higher is more urgent): 1
Enter the arrival time for pid 5: 3
Enter the burst time for pid 5: 7
Enter priorty (higher is more urgent): 4


'1' '1' '2' '2' '5' '5' '5' '5' '5' '5' '5' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '3' '3' '3' '3' '3' '4'
avg turnaround is: 16
avg wait is: 44
final results:
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
1               0               15              24              24              3
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
2               2               2               2               4               5
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
5               3               7               8               11              4
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
3               4               5               25              29              2
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
4               7               1               23              30              1
hariketsheth@ubuntu:~/Desktop/lab5$
```

ii. Pre-emptive

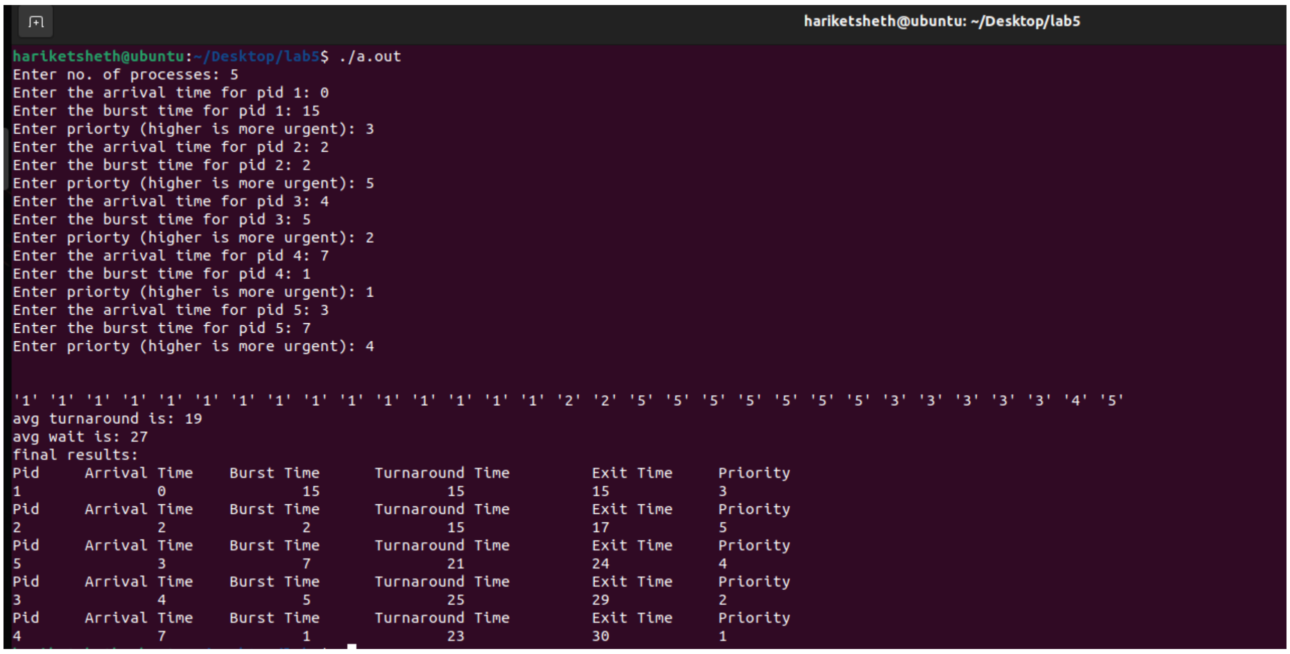


```cpp
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;

struct process {
    int pid;
    int arrtime;
    int burstime;
    int remaintime;
    int priority; //higher is more urgent
    int wtime = 999;
    int turnaroundtime;
    int exitime;
};
bool compareArr(process p1, process p2) {
    return (p1.arrtime > p2.arrtime);
}

bool compareRem(process p1, process p2) {
    return (p1.remaintime > p2.remaintime);
}

bool comparePri(process p1, process p2) {
    return (p1.priority < p2.priority);
}

int main() {
    int n;
    int avgturnaround = 0, avgwait = 0;
    cout << "Enter no. of processes: ";
    cin >> n;
    int curtime = 0, ttlexec = 0;
    vector < process > notinqueue, inqueue, completed;
```



```
    for (int i = 1; i < n + 1; i++) {
        process a;
        a.pid = i;
        cout << "Enter the arrival time for pid " << i << ": ";
        cin >> a.arrtime;
        cout << "Enter the burst time for pid " << i << ": ";
        cin >> a.burstime;
        a.remaintime = a.burstime;
        cout << "Enter priorty (higher is more urgent): ";
        cin >> a.priority;
        ttlexec = ttlexec + a.burstime;
        notinqueue.push_back(a);
    }
    cout << endl << endl;
    sort(notinqueue.begin(), notinqueue.end(), compareArr);
    for (curtime; curtime <= ttlexec; curtime++) {
        int k1 = 999;
        if (!notinqueue.empty())
            k1 = notinqueue.back().arrtime;

        if (inqueue.empty()) {
            if (k1 == curtime) {
                process a = notinqueue.back();
                notinqueue.pop_back();
                inqueue.push_back(a);
                inqueue.back().remaintime--;
                cout << "'" << inqueue.back().pid << "' ";
                inqueue.back().wtime = 0;
                avgwait = avgwait + curtime;
                if (inqueue.back().remaintime == 0) {
                    inqueue.back().exitime = curtime + 1;
                    inqueue.back().turnaroundtime = inqueue.back().exitime -
inqueue.back().arrtime;
                    avgturnaround = avgturnaround + inqueue.back().turnaroundtime;
                    process a = inqueue.back();
                    completed.push_back(a);
                    inqueue.pop_back();
                }
            }
        } else {
            if (k1 == curtime) {
                process a = notinqueue.back();
                notinqueue.pop_back();
                inqueue.push_back(a);
            }

            sort(inqueue.begin(), inqueue.end(), comparePri); //for primitive

            if (!inqueue.empty()) {
                inqueue.back().remaintime--;
                cout << "'" << inqueue.back().pid << "' ";
            }
            if (inqueue.back().wtime != 999) {
                inqueue.back().wtime = curtime;
                avgwait = avgwait + curtime;
            }
            if (inqueue.back().remaintime == 0) {
                inqueue.back().exitime = curtime + 1;
                inqueue.back().turnaroundtime = inqueue.back().exitime -
inqueue.back().arrtime;
                avgturnaround = avgturnaround + inqueue.back().turnaroundtime;
                process a = inqueue.back();
                completed.push_back(a);
                inqueue.pop_back();
                sort(inqueue.begin(), inqueue.end(), comparePri); //for primitive
            }
        }
    }

    avgturnaround = avgturnaround / n;
    avgwait = avgwait / n;
    cout << "\navg turnaround is: " << avgturnaround;
    cout << "\navg wait is: " << avgwait << endl;
    cout << "final results: " << endl;
    sort(completed.begin(), completed.end(), compareArr);
    for (int i = 0; i < n; i++) {
        cout << "Pid\tArrival Time\tBurst Time\tTurnaround Time\t\tExit 
Time\tPriority\n";
        cout << completed.back().pid << "\t\t" << completed.back().arrtime << 
"\t\t" <<
            completed.back().burstime << "\t\t" << completed.back().turnaroundtime 
<< "\t\t" <<
            completed.back().exitime << "\t\t" << completed.back().priority << 
"\n";
        completed.pop_back();
    }
    return 0;
}
```

OUTPUT:

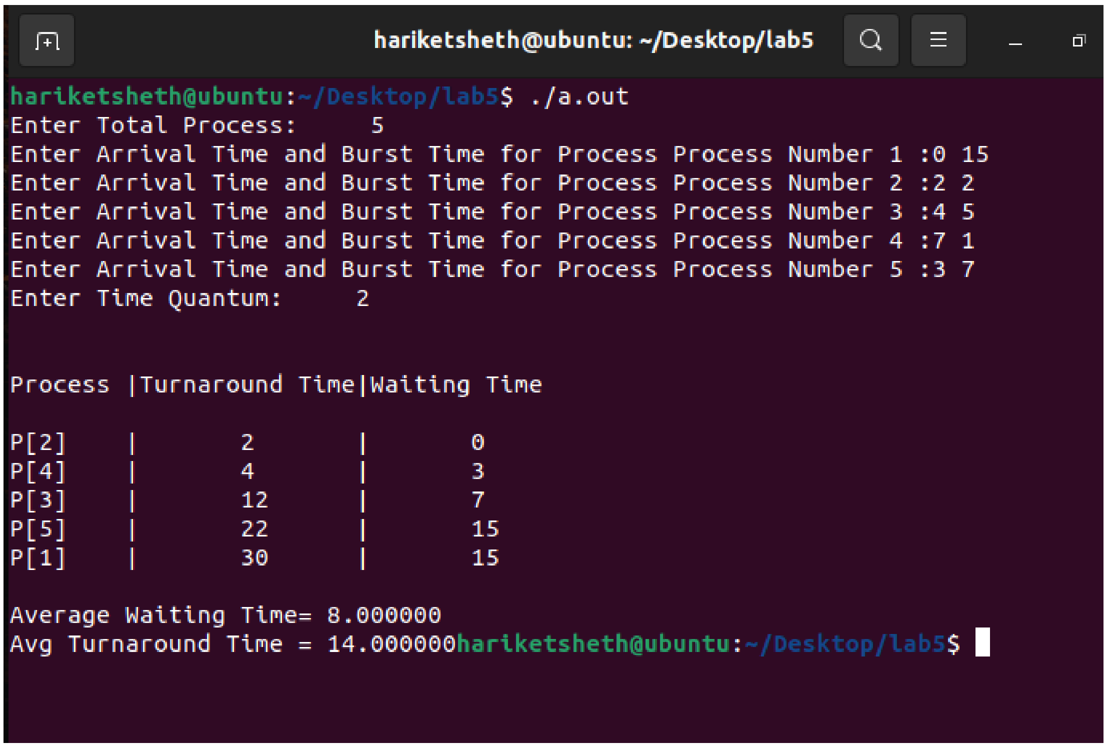


```
hariketsheth@ubuntu:~/Desktop/lab5$ ./a.out
Enter no. of processes: 5
Enter the arrival time for pid 1: 0
Enter the burst time for pid 1: 15
Enter priorty (higher is more urgent): 3
Enter the arrival time for pid 2: 2
Enter the burst time for pid 2: 2
Enter priorty (higher is more urgent): 5
Enter the arrival time for pid 3: 4
Enter the burst time for pid 3: 5
Enter priorty (higher is more urgent): 2
Enter the arrival time for pid 4: 7
Enter the burst time for pid 4: 1
Enter priorty (higher is more urgent): 1
Enter the arrival time for pid 5: 3
Enter the burst time for pid 5: 7
Enter priorty (higher is more urgent): 4


'1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '1' '2' '2' '5' '5' '5' '5' '5' '5' '5' '3' '3' '3' '3' '3' '4' '5'
avg turnaround is: 19
avg wait is: 27
final results:
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
1               0               15              15              15              3
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
2               2               2               15              17              5
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
5               3               7               21              24              4
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
3               4               5               25              29              2
Pid     Arrival Time    Burst Time      Turnaround Time         Exit Time       Priority
4               7               1               23              30              1
```

e. Round Robin

```
#include<stdio.h>

int main()
{

    int count,j,n,time,remain,flag=0,time_quantum;
    int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
    printf("Enter Total Process:\t ");
    scanf("%d",&n);
    remain=n;
    for(count=0;count<n;count++)
    {
            printf("Enter Arrival Time and Burst Time for Process Process Number %d
:",count+1);
            scanf("%d",&at[count]);
            scanf("%d",&bt[count]);
            rt[count]=bt[count];
    }
    printf("Enter Time Quantum:\t");
    scanf("%d",&time_quantum);
    printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
    for(time=0,count=0;remain!=0;)
    {
            if(rt[count]<=time_quantum && rt[count]>0)
            {
                time+=rt[count];
                rt[count]=0;
                flag=1;
            }
            else if(rt[count]>0)
            {
                rt[count]-=time_quantum;
                time+=time_quantum;
            }
            if(rt[count]==0 && flag==1)
            {
                remain--;
                printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-
bt[count]);
                wait_time+=time-at[count]-bt[count];
                turnaround_time+=time-at[count];
                flag=0;
            }
            if(count==n-1)
            count=0;
            else if(at[count+1]<=time)
            count++;
            else
            count=0;
    }
    printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
    printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);
        return 0;
}
```

OUTPUT:



```
hariketsheth@ubuntu:~/Desktop/lab5$ ./a.out
Enter Total Process:     5
Enter Arrival Time and Burst Time for Process Process Number 1 :0 15
Enter Arrival Time and Burst Time for Process Process Number 2 :2 2
Enter Arrival Time and Burst Time for Process Process Number 3 :4 5
Enter Arrival Time and Burst Time for Process Process Number 4 :7 1
Enter Arrival Time and Burst Time for Process Process Number 5 :3 7
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[2]    |       2       |       0
P[4]    |       4       |       3
P[3]    |       12      |       7
P[5]    |       22      |       15
P[1]    |       30      |       15

Average Waiting Time= 8.000000
Avg Turnaround Time = 14.000000hariketsheth@ubuntu:~/Desktop/lab5$ 
```

FCFS, Shortest Remaining Time First and Round Robin Scheduling algorithms have performed well in this scenario with **less average waiting time** and **more turnaround time**