Nithin S
221IT085

IT253 Lab Assignment 3

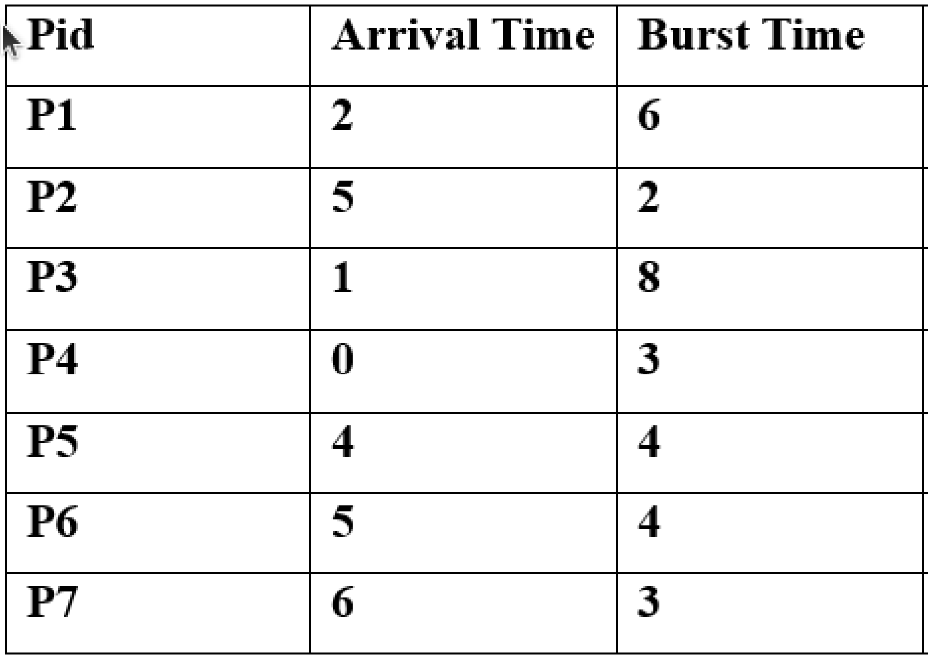
Simulation of CPU Scheduling
(Non-Preemptive)

First Come First Server [FCFS]

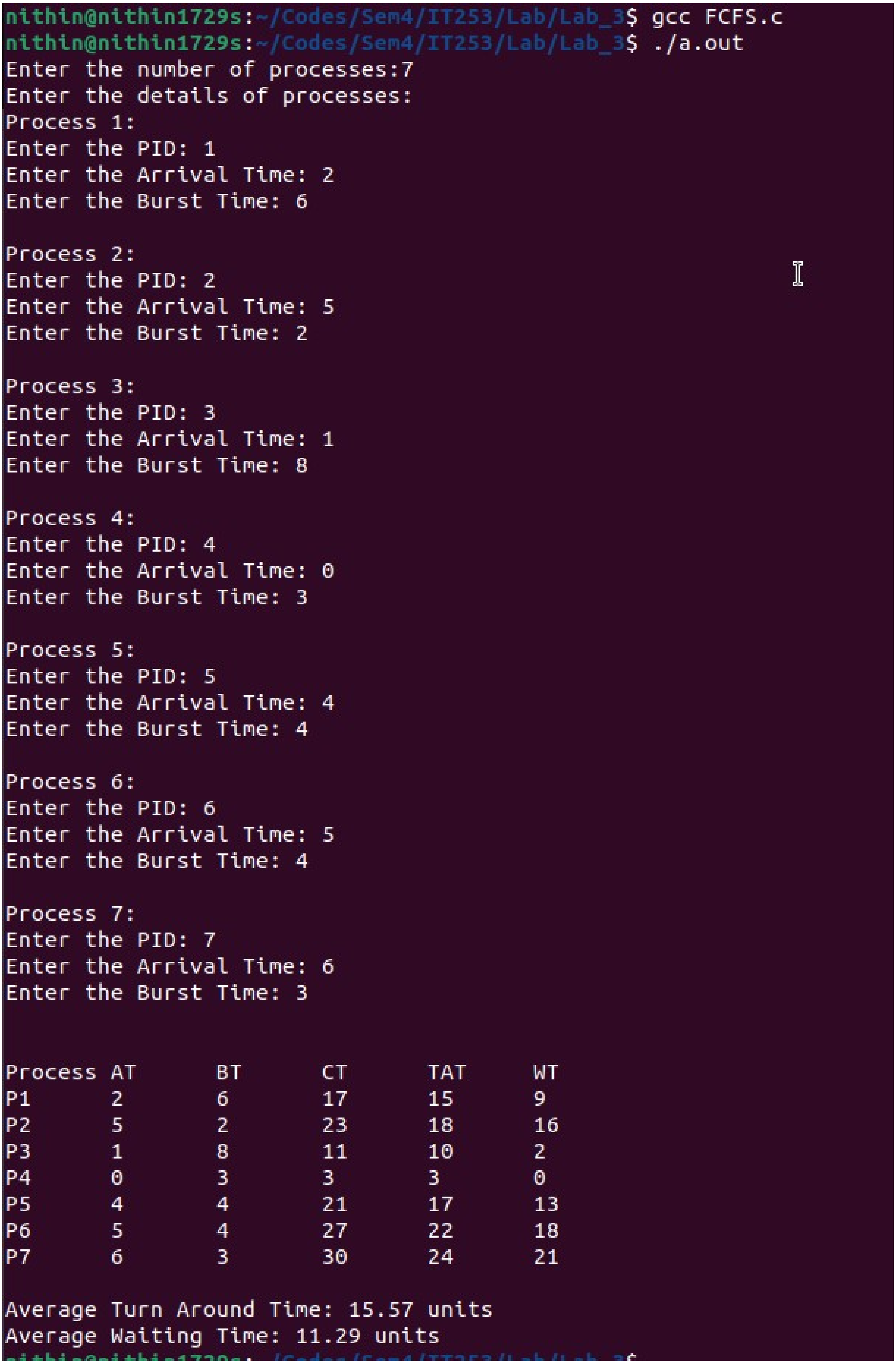
CODE

```
#include <stdio.h>

struct Process
{
    int pid;
    int AT;
    int BT;
    int CT;
    int TAT;
    int WT;
};

void sortOnAT(struct Process *processes, int size)
{
    int i, j;
    struct Process tmp;

    for (i = 0; i < size - 1; i++)
    {
        for (j = 0; j < size - i - 1; j++)
        {
            if (processes[j].AT > processes[j +
1].AT)
            {

                tmp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = tmp;
            }
        }
    }
}

void sortOnPID(struct Process *processes, int size)
{
    int i, j;
    struct Process tmp;

    for (i = 0; i < size - 1; i++)
    {
        for (j = 0; j < size - i - 1; j++)
        {
            if (processes[j].pid > processes[j +
1].pid)
            {

                tmp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = tmp;
            }
        }
    }
}

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

    printf("Enter the details of processes:\n");

    for (int i = 0; i < n; i++)
    {
        printf("Process %d:\n", i + 1);
        printf("Enter the PID: ");
        scanf("%d", &processes[i].pid);
        printf("Enter the Arrival Time: ");
        scanf("%d", &processes[i].AT);
        printf("Enter the Burst Time: ");
        scanf("%d", &processes[i].BT);
        printf("\n");
    }

    sortOnAT(processes, n);

    int ct_previous = 0;
    ct_previous += processes[0].AT;
    int avg_TAT = 0;
    int avg_WT = 0;

    for (int i = 0; i < n; i++)
    {
        ct_previous += processes[i].BT;
        processes[i].CT = ct_previous;
        processes[i].TAT = processes[i].CT -
processes[i].AT;
        processes[i].WT = processes[i].TAT -
processes[i].BT;
        if (ct_previous < processes[i + 1].AT)
        {
            int t = processes[i + 1].AT -
ct_previous;
            ct_previous += t;
        }
        avg_TAT += processes[i].TAT;
        avg_WT += processes[i].WT;
    }
    sortOnPID(processes, n);
    printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++)
    {
        printf("%d\t%d\t%d\t%d\t%d\t%d\n",
processes[i].pid, processes[i].AT, processes[i].BT,
               processes[i].CT, processes[i].TAT,
processes[i].WT);
    }

    printf("\nAverage Turn Around Time: %.2f",
(float)avg_TAT / n);
    printf("\nAverage Waiting Time: %.2f",
(float)avg_WT / n);

    return 0;
}
```

OUTPUT



Pid	Arrival Time	Burst Time
P1	2	6
P2	5	2
P3	1	8
P4	0	3
P5	4	4
P6	5	4
P7	6	3



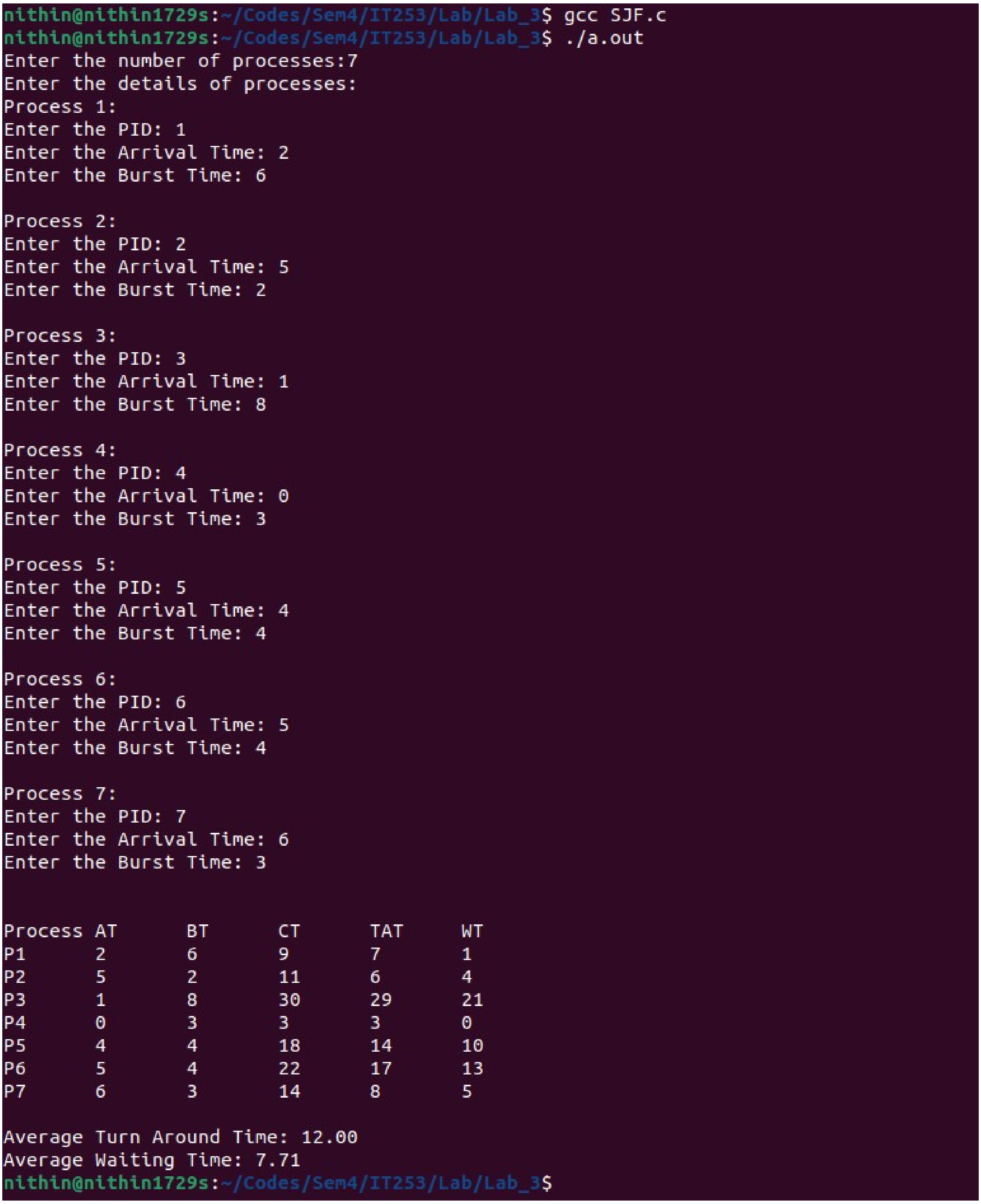
```
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ gcc FCFS.c
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ ./a.out
Enter the number of processes:7
Enter the details of processes:
Process 1:
Enter the PID: 1
Enter the Arrival Time: 2
Enter the Burst Time: 6

Process 2:
Enter the PID: 2
Enter the Arrival Time: 5
Enter the Burst Time: 2

Process 3:
Enter the PID: 3
Enter the Arrival Time: 1
Enter the Burst Time: 8

Process 4:
Enter the PID: 4
Enter the Arrival Time: 0
Enter the Burst Time: 3

Process 5:
Enter the PID: 5
Enter the Arrival Time: 4
Enter the Burst Time: 4

Process 6:
Enter the PID: 6
Enter the Arrival Time: 5
Enter the Burst Time: 4

Process 7:
Enter the PID: 7
Enter the Arrival Time: 6
Enter the Burst Time: 3


Process AT      BT      CT      TAT     WT
P1      2       6       17      15      9
P2      5       2       23      18      16
P3      1       8       11      10      2
P4      0       3       3       3       0
P5      4       4       21      17      13
P6      5       4       27      22      18
P7      6       3       30      24      21

Average Turn Around Time: 15.57 units
Average Waiting Time: 11.29 units
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$
```

Shortest Job First [SJF]

CODE

```
#include <stdio.h>
#include<stdbool.h>

struct Process
{
    int pid;
    int AT;
    int BT;
    int Priority;
    int CT;
    int TAT;
    int WT;
    bool visited;
};

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

    struct Process processes[n];

    printf("Enter the details of processes:\n");

    for (int i = 0; i < n; i++)
    {
        printf("Process %d:\n", i + 1);
        printf("Enter the PID: ");
        scanf("%d", &processes[i].pid);
        printf("Enter the Arrival Time: ");
        scanf("%d", &processes[i].AT);
        printf("Enter the Burst Time: ");
        scanf("%d", &processes[i].BT);
        processes[i].visited=false;
        printf("\n");
    }

    int i,j;
    int minCT=0;
    for(j=0;j<n;)
    {
        int minBT,idx;
        minBT=idx=1e9;
        for(int i=0;i<n;i++)
        {
            if(processes[i].visited==false)
            {
                if(minCT>=processes[i].AT &&
minBT>processes[i].BT)
                {
                    minBT=processes[i].BT;
                    idx=i;
                }
                else if(minCT>=processes[i].AT &&
minBT==processes[i].BT)
                {

if(processes[idx].AT>processes[i].AT)
                    {
                        minBT=processes[i].BT;
                        idx=i;
                    }
                }
            }
        }

        if(idx==1e9)
        {
            minCT++;
        }
        else
        {
            minCT+=minBT;
            processes[idx].visited=true;
            processes[idx].CT=minCT;
            j++;
        }
    }
    int avg_TAT=0;
    int avg_WT=0;
    for(int i=0;i<n;i++)
    {
        processes[i].TAT=processes[i].CT-
processes[i].AT;
        processes[i].WT=processes[i].TAT-
processes[i].BT;
        avg_TAT+=processes[i].TAT;
        avg_WT+=processes[i].WT;
    }


    printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n",
processes[i].pid, processes[i].AT, processes[i].BT,
               processes[i].CT, processes[i].TAT,
processes[i].WT);
    }

    printf("\nAverage Turn Around Time: %.2f",
(float)avg_TAT / n);
    printf("\nAverage Waiting Time: %.2f\n",
(float)avg_WT / n);

    return 0;
}
```

OUTPUT



```
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ gcc SJF.c
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ ./a.out
Enter the number of processes:7
Enter the details of processes:
Process 1:
Enter the PID: 1
Enter the Arrival Time: 2
Enter the Burst Time: 6

Process 2:
Enter the PID: 2
Enter the Arrival Time: 5
Enter the Burst Time: 2

Process 3:
Enter the PID: 3
Enter the Arrival Time: 1
Enter the Burst Time: 8

Process 4:
Enter the PID: 4
Enter the Arrival Time: 0
Enter the Burst Time: 3

Process 5:
Enter the PID: 5
Enter the Arrival Time: 4
Enter the Burst Time: 4

Process 6:
Enter the PID: 6
Enter the Arrival Time: 5
Enter the Burst Time: 4

Process 7:
Enter the PID: 7
Enter the Arrival Time: 6
Enter the Burst Time: 3


Process AT      BT      CT      TAT     WT
P1      2       6       9       7       1
P2      5       2       11      6       4
P3      1       8       30      29      21
P4      0       3       3       3       0
P5      4       4       18      14      10
P6      5       4       22      17      13
P7      6       3       14      8       5

Average Turn Around Time: 12.00
Average Waiting Time: 7.71
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$
```

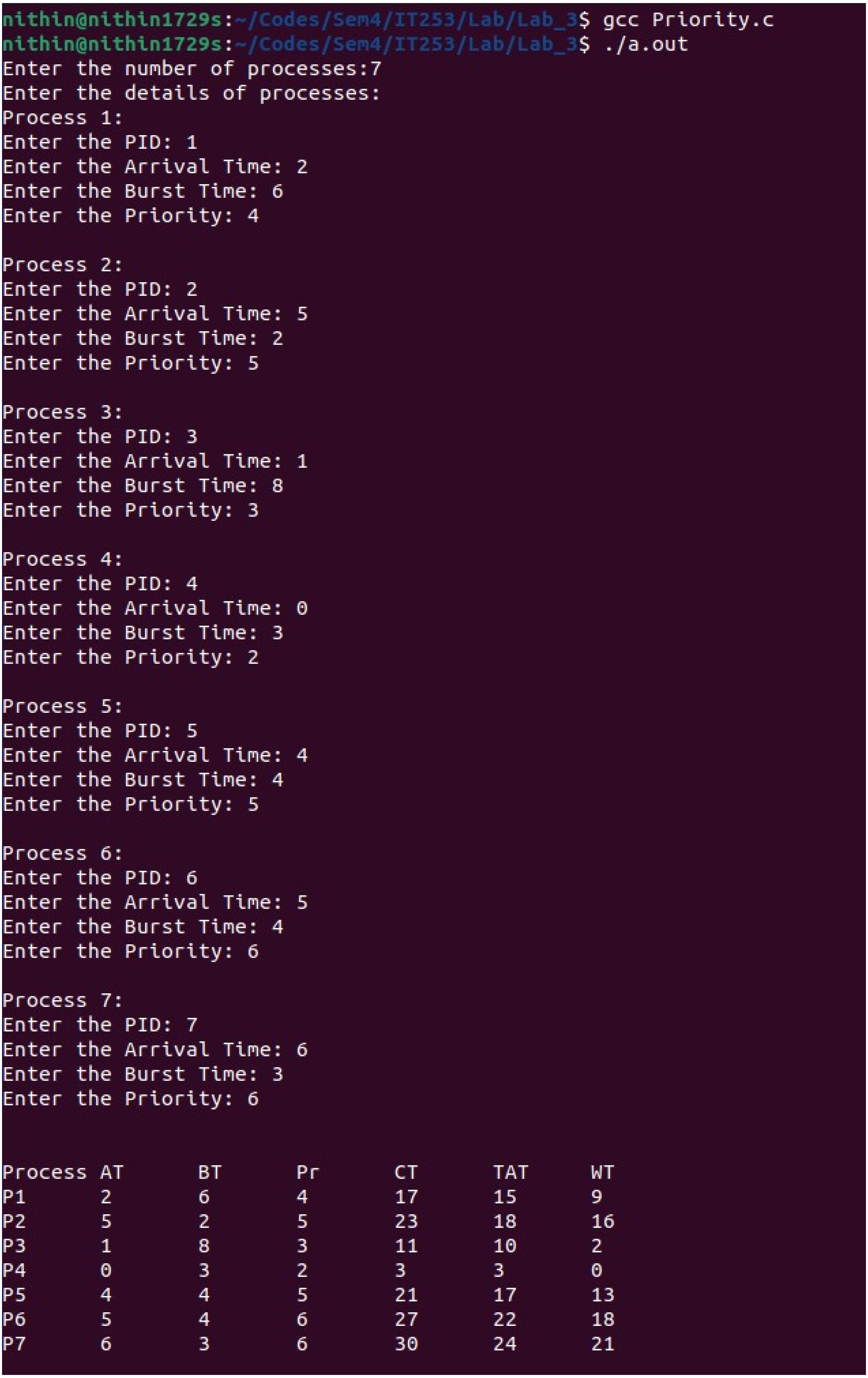
Priority

CODE

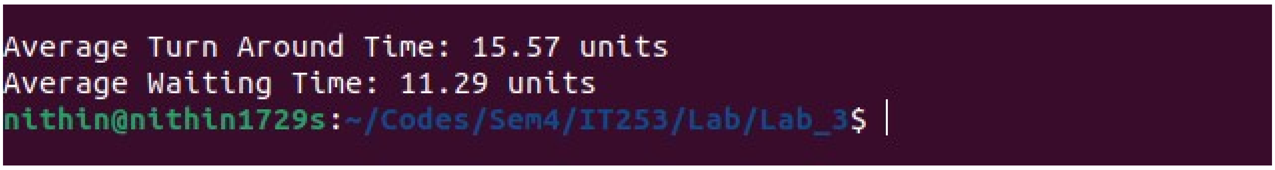
```
#include <stdio.h>
#include<stdbool.h>

struct Process
{
    int pid;
    int AT;
    int BT;
    int Priority;
    int CT;
    int TAT;
    int WT;
    bool visited;
};

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

    struct Process processes[n];

    printf("Enter the details of processes:\n");

    for (int i = 0; i < n; i++)
    {
        printf("Process %d:\n", i + 1);
        printf("Enter the PID: ");
        scanf("%d", &processes[i].pid);
        printf("Enter the Arrival Time: ");
        scanf("%d", &processes[i].AT);
        printf("Enter the Burst Time: ");
        scanf("%d", &processes[i].BT);
        printf("Enter the Priority: ");
        scanf("%d", &processes[i].Priority);
        processes[i].visited=false;
        printf("\n");
    }

    int i,j;
    int minCT=0;
    for(j=0;j<n;)
    {
        int minBT,idx;
        minBT=idx=1e9;
        for(i=0;i<n;i++)
        {
            if(processes[i].visited==false)
            {
                if(minCT>=processes[i].AT &&
minBT>processes[i].Priority)
                {
                    minBT=processes[i].Priority;
                    idx=i;
                }
                else if(minCT>=processes[i].AT &&
minBT==processes[i].Priority)
                {

if(processes[idx].AT>processes[i].AT)
                    {
                        minBT=processes[i].Priority;
                        idx=i;
                    }
                }
            }
        }

        if(idx==1e9)
        {
            minCT++;
        }
        else
        {
            minCT+=processes[idx].BT;
            processes[idx].visited=true;
            processes[idx].CT=minCT;
            j++;
        }
    }
    int avg_TAT=0;
    int avg_WT=0;
    for(int i=0;i<n;i++)
    {
        processes[i].TAT=processes[i].CT-
processes[i].AT;
        processes[i].WT=processes[i].TAT-
processes[i].BT;
        avg_TAT+=processes[i].TAT;
        avg_WT+=processes[i].WT;
    }


    printf("\nProcess\tAT\tBT\tPr\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\n",
processes[i].pid, processes[i].AT,
processes[i].BT,processes[i].Priority,
               processes[i].CT, processes[i].TAT,
processes[i].WT);
    }

    printf("\nAverage Turn Around Time: %.2f units",
(float)avg_TAT / n);
    printf("\nAverage Waiting Time: %.2f units\n",
(float)avg_WT / n);

    return 0;
}
```

OUTPUT

Pid	Arrival Time	Burst Time	Priority
P1	2	6	4
P2	5	2	5
P3	1	8	3
P4	0	3	2
P5	4	4	5
P6	5	4	6
P7	6	3	6



```
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ gcc Priority.c
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ ./a.out
Enter the number of processes:7
Enter the details of processes:
Process 1:
Enter the PID: 1
Enter the Arrival Time: 2
Enter the Burst Time: 6
Enter the Priority: 4

Process 2:
Enter the PID: 2
Enter the Arrival Time: 5
Enter the Burst Time: 2
Enter the Priority: 5

Process 3:
Enter the PID: 3
Enter the Arrival Time: 1
Enter the Burst Time: 8
Enter the Priority: 3

Process 4:
Enter the PID: 4
Enter the Arrival Time: 0
Enter the Burst Time: 3
Enter the Priority: 2

Process 5:
Enter the PID: 5
Enter the Arrival Time: 4
Enter the Burst Time: 4
Enter the Priority: 5

Process 6:
Enter the PID: 6
Enter the Arrival Time: 5
Enter the Burst Time: 4
Enter the Priority: 6

Process 7:
Enter the PID: 7
Enter the Arrival Time: 6
Enter the Burst Time: 3
Enter the Priority: 6


Process AT      BT      Pr      CT      TAT     WT
P1      2       6       4       17      15      9
P2      5       2       5       23      18      16
P3      1       8       3       11      10      2
P4      0       3       2       3       3       0
P5      4       4       5       21      17      13
P6      5       4       6       27      22      18
P7      6       3       6       30      24      21
```



```
Average Turn Around Time: 15.57 units
Average Waiting Time: 11.29 units
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_3$ 
```