**Student Name:** Divyansh Kumar

**Student ID:** 11814920 Roll No: 36 Section: K18HV

**Email Address:** divyansh006kumar@gmail.com

**GitHub Link:** <https://github.com/divyansh006/cse316>

**Question:**

Ques. 14. Write a program to implement priority scheduling algorithm with context switching time. Prompt to user to enter the number of processes and then enter their priority, burst time and arrival time also. Now whenever operating system preempts a process and shifts CPU’s control to some another process of higher priority assume that it takes 2 seconds for context switching(dispatcher latency).Form a scenario, where we can give the processes are assigned with priority where the lower integer number is higher priority and then context switch .. as the process waits the priority of the process increase at rate of one per 2-time units of wait.

Calculate waiting time and turnaround time for each process.

**1. Description of the problem:**

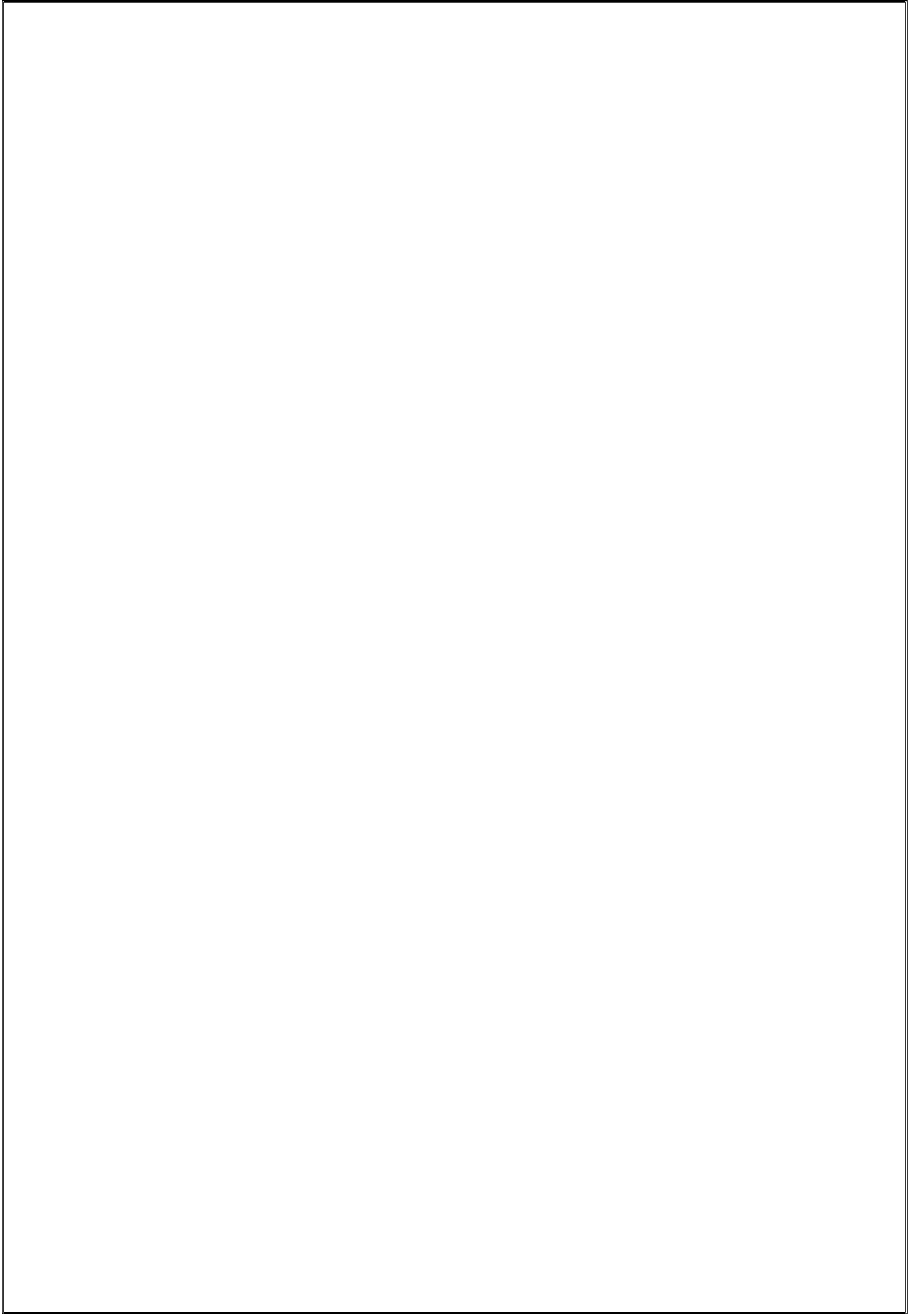
Here we have to take the number of process from the user followed by their priority, arrival time and burst time after that we have to apply the concept of priority scheduling. Here we follow the ‘Lower the Number Higher the Priority’ concept to apply the scheduling.

We have to take the dispatch latency as 2 unit time i.e. the time taken to switch the processor from one process to another. And also the priority of each process increases one unit with the wait-time of 2 unit (Here the priority number will decrease one unit as we are using ‘Lower the Number Higher the Priority’ concept).

Ex.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Priority** | **AT** | **BT** | **CT** | **TAT***(=CT-AT)* | **WT***(=TAT-BT)* |
| **P0** | **40** | **0** | **5** | **31** | 31-0= **31** | 31-5= **26** |
| **P1** | **30** | **1** | **4** | **25** | 25-1= **24** | 24-4= **20** |
| **P2** | **20** | **2** | **2** | **20** | 20-2= **18** | 18-2= **16** |
| **P3** | **10** | **4** | **10** | **16** | 16-4= **12** | 12-10= **2** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Gantt Chart | |  |  |  |  |  |  |  |  |  |  |  |
| **P0** | DL | **P1** | DL | **P3** | DL |  | **P2** | DL | **P1** | DL |  | **P0** |
| 0 | 1 | 3 | 4 | 6 | 16 | 18 | 20 |  | 22 | 25 | 27 | 31 |

**2. Algorithm:**

**ALGORITHM Priority Scheduling:**

Initialize, t=0, complete=0

Input: n //n is number Of Process

Initialize an two dimensional array of size (8\*n)

|  |  |  |  |
| --- | --- | --- | --- |
| arr[8][n]; | |  |  |
| //row-0 is | | ProcessNo | |
| //row-1 is | | ArrivalTime | |
| //row-2 is | | Priority |  |
| //row-3 is | | BurstTime | |
| //row-4 | is | WaitTime |  |
| //row-5 | is | State | 0-idle, (-1)-completed, 2-queue; |
| //row-6 | is | CompletionTime | |
| //row-7 | is | WTIP | WaitTimeToIncreasePriority |

Input: ArrivalTime for each process

Priority for each Process

BurstTime for each Process

Initialize: WaitTime=0, State=0, CompletionTime=0, WTIP=0 for each process.

for i= 0...n

if ArrivalTime[i]==0

minPriIndexC = i;

break;

//end of if statement

//end of for loop

Repeat while complete==0

for i=0...n

if ArrivalTime[i]==t

Set State=2

//end of if statement

//end of for loop

t++ //Incrimenting t by 1

Set minPri = q=100000, minPriIndex=0

//Finding the minimum priority

for i=0...n

if Priority[i]<minPri && State[i]==2

minPriIndex=i

minPri=Priority[i]

//end of if statement

//end of for loop

//Dispatch Latency

if minPriIndexC != minPriIndex

Set time=2

Repeat while time !=0

for i=0...n

if ArrivalTime[i]==t

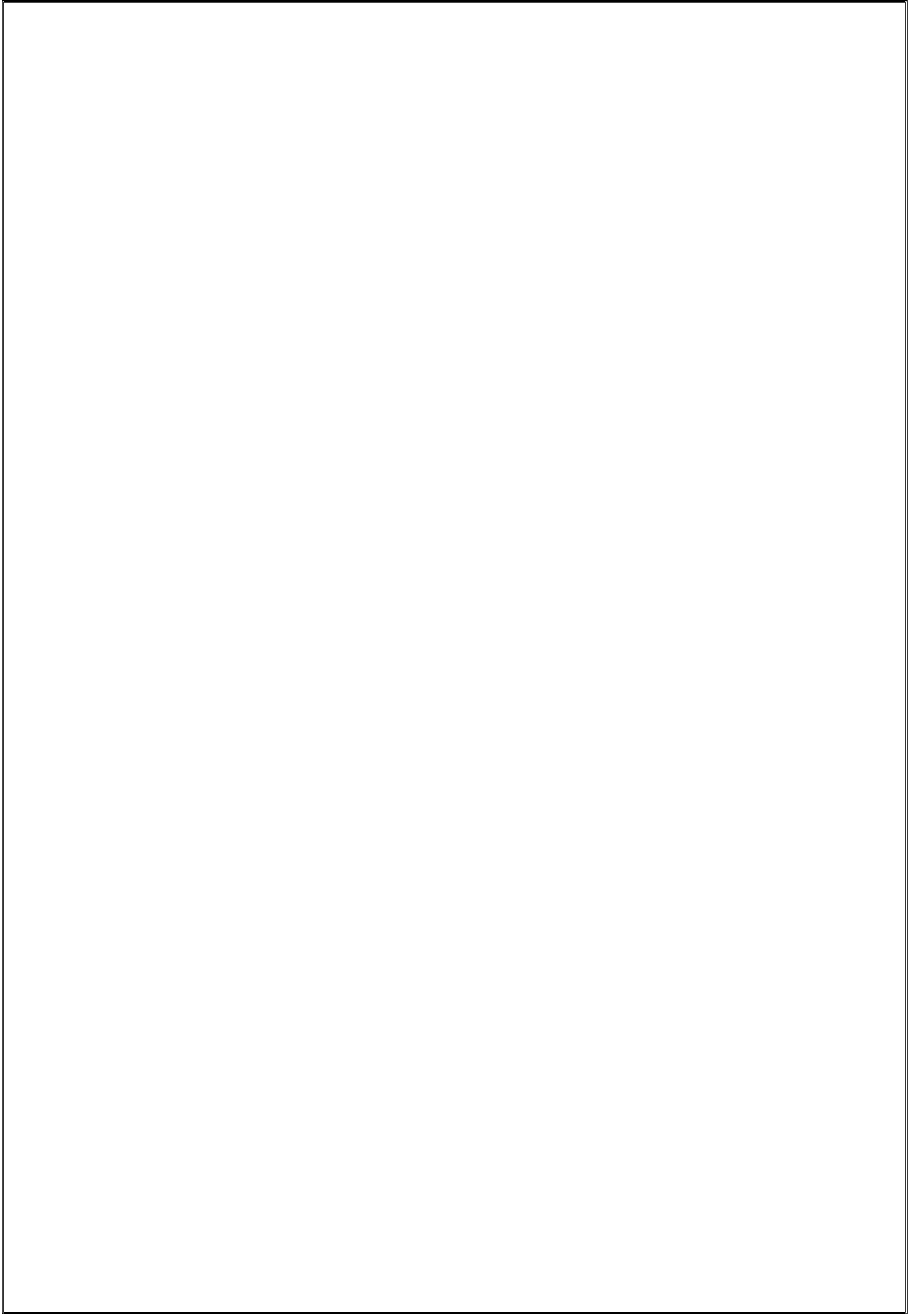
State[i]=2

//end of if statement

//end of for loop

t++

//Incriment t by 1 unit time

for i=0...n

WaitTime[i] +=1

WTIP[i] +=1

if WTIP[i]==2:

WTIP[i]=0

Priority[i] -=1

//end of if statement

//end of for loop

time-- //Decrimenting time by 1

End of inner while loop

//end of if statement

//Execution Section

minPriIndexC = minPriIndex

BurstTime[minPriIndex] -=1

WaitTime[minPriIndex] -=1

WTIP[minPriIndex] -=1

if BurstTime[minPriIndex]==0

CompletionTime[minPriIndex] = t

State[minPriIndex] = -1

WTIP[minPriIndex] = 0

//end of if statement

//Checking for the completion of the processes

Set temp=0

for i=0...n

temp = temp + State[i]

//end of for loop

if (temp\*(-1))==n

complete = -1

//end of if statement

//end of while loop

**3. Complexity:**

Total Complexity:

* 43\*O(1) + 31\*O(n) + 2\*log(n) + 17\*O(m) + 4\*O(mn) + 4\*O(mlogn) + 4\*O(3m) + 3\*O(3mlogn) + 2\*(3m(n/3) + 3\*O(m/3) + 2\*(m/9)

Overall Complexity: O(mn)

Where, m = Turn Around Time

n = Number of process

The Code for the problem along with their respective complexity of each line

#include<stdio.h>

#include<stdlib.h>

//Symbol for complexity: n = Total no. of process

// m = Turn Around Time

int main(){

int noOfProcess;

int i, at, pr, bt;

int t=0;

int complete=0;

int minPriIndexC;

int minPri, minPriIndex;

//Taking User Input

printf("No of process: ");

scanf("%d",&noOfProcess);

int arr[8][noOfProcess];

for(i=0; i<noOfProcess; i++){

printf("\nFor Process P%d \n", i);

printf("Priority: ");

scanf("%d", &pr);

printf("Arrival Time: ");

scanf("%d", &at);

printf("Burst Time: ");

scanf("%d", &bt);

arr[0][i]=i;

arr[1][i]=at; //Arrival Time

arr[2][i]=pr; //Priority

arr[3][i]=bt; //Burst Time

arr[4][i]=0; //Wait Time

arr[5][i]=0; //State

arr[6][i]=0; //Turn Around Time

arr[7][i]=0; //WaitTime to incriment Priority

}

printf("\n-----------------------------------\n\n");

//Saving the burst time of each process.

int burstTime[noOfProcess];

for(i=0; i<noOfProcess;i++){

burstTime[i]=arr[3][i];

}

int initialPri[noOfProcess];

for(i=0; i<noOfProcess;i++){

initialPri[i]=arr[2][i];

}

//Checking initial minimum priority.

for(i=0; i<noOfProcess; i++){

if(arr[1][i]==0){

minPriIndexC = i;

break;

}

}

while(complete==0){

//Checking for the arrival time

for(i=0;i<noOfProcess; i++){

if(arr[1][i]==t){

arr[5][i]=2;

}

}

minPri = 100000;

minPriIndex = 0;

//Finding the minimum priority

for(i=0; i<noOfProcess; i++){

if(arr[2][i]<minPri&&arr[5][i]==2){

minPriIndex = i;

minPri = arr[2][i];

}

}

t++;

if(minPriIndexC!=minPriIndex){

int time=2;

while(time!=0){

//Checking for the arrival time

for(i=0;i<noOfProcess; i++){

if(arr[1][i]==t){

arr[5][i]=2;

}

}

t++;

for(i=0; i<noOfProcess;i++){

if(arr[5][i]==2){

arr[4][i] +=1;

arr[7][i] +=1;

if(arr[7][i]==2){

arr[7][i]=0;

arr[2][i] -=1;

}

}

}

time--;

}

}

//Execution

minPriIndexC = minPriIndex;

arr[3][minPriIndex] -=1;

arr[4][minPriIndex] -=1;

arr[7][minPriIndex] -=1;

if(arr[3][minPriIndex]==0){

arr[6][minPriIndex] = t; //--

arr[5][minPriIndex] = -1;

arr[7][minPriIndex] = 0; //--

}

//Checking for the completed process.

for(i=0; i<noOfProcess;i++){

if(arr[5][i]==2){

arr[4][i] +=1;

arr[7][i] +=1;

if(arr[7][i]==2){

arr[7][i]=0;

arr[2][i] -=1;

}

}

}

//Checking the completion

int temp=0;

for(i=0; i<noOfProcess;i++){

temp=temp+arr[5][i];

}

if((temp\*(-1))==noOfProcess){

complete=-1;

}

}

/\*End of the logic\*/

//Output

printf("Priority Scheduling Table: \n");

printf("\nProcNo: \t");

for(i=0; i<noOfProcess; i++){

printf("P%d\t", arr[0][i]);

}

printf("\nArrT: \t");

for(i=0; i<noOfProcess; i++){

printf("%d\t", arr[1][i]);

}

printf("\nInitPri: \t"); for(i=0; i<noOfProcess; i++){

printf("%d\t", initialPri[i]);

}

printf("\nBurstT: \t");

for(i=0; i<noOfProcess; i++){

printf("%d\t", burstTime[i]);

}

printf("\nWaitT: \t"); for(i=0; i<noOfProcess; i++){

printf("%d\t", (arr[6][i]-arr[1][i])-burstTime[i]); }

printf("\nCompT: \t");

for(i=0; i<noOfProcess; i++){

printf("%d\t", arr[6][i]);

}

printf("\nTAT: \t");

for(i=0; i<noOfProcess; i++){

printf("%d\t", arr[6][i]-arr[1][i]); }

printf("\nFinalPri: \t");

for(i=0; i<noOfProcess; i++){

printf("%d\t", arr[2][i]);

}

printf("\n\n\nRequired Answers: ");

printf("\n\nPNo: |\t");

for(i=0; i<noOfProcess; i++){

printf(" P%d", arr[0][i]);

printf(" | ");

}

printf("\n\nTAT: |\t");

for(i=0; i<noOfProcess; i++){

printf("%5d", arr[6][i]-arr[1][i]);

printf(" | ");

}

printf("\n\nWT: |\t");

int totalWaitTime=0;

for(i=0; i<noOfProcess; i++){

printf("%5d", (arr[6][i]-arr[1][i])-burstTime[i]);

totalWaitTime += (arr[6][i]-arr[1][i])-burstTime[i];

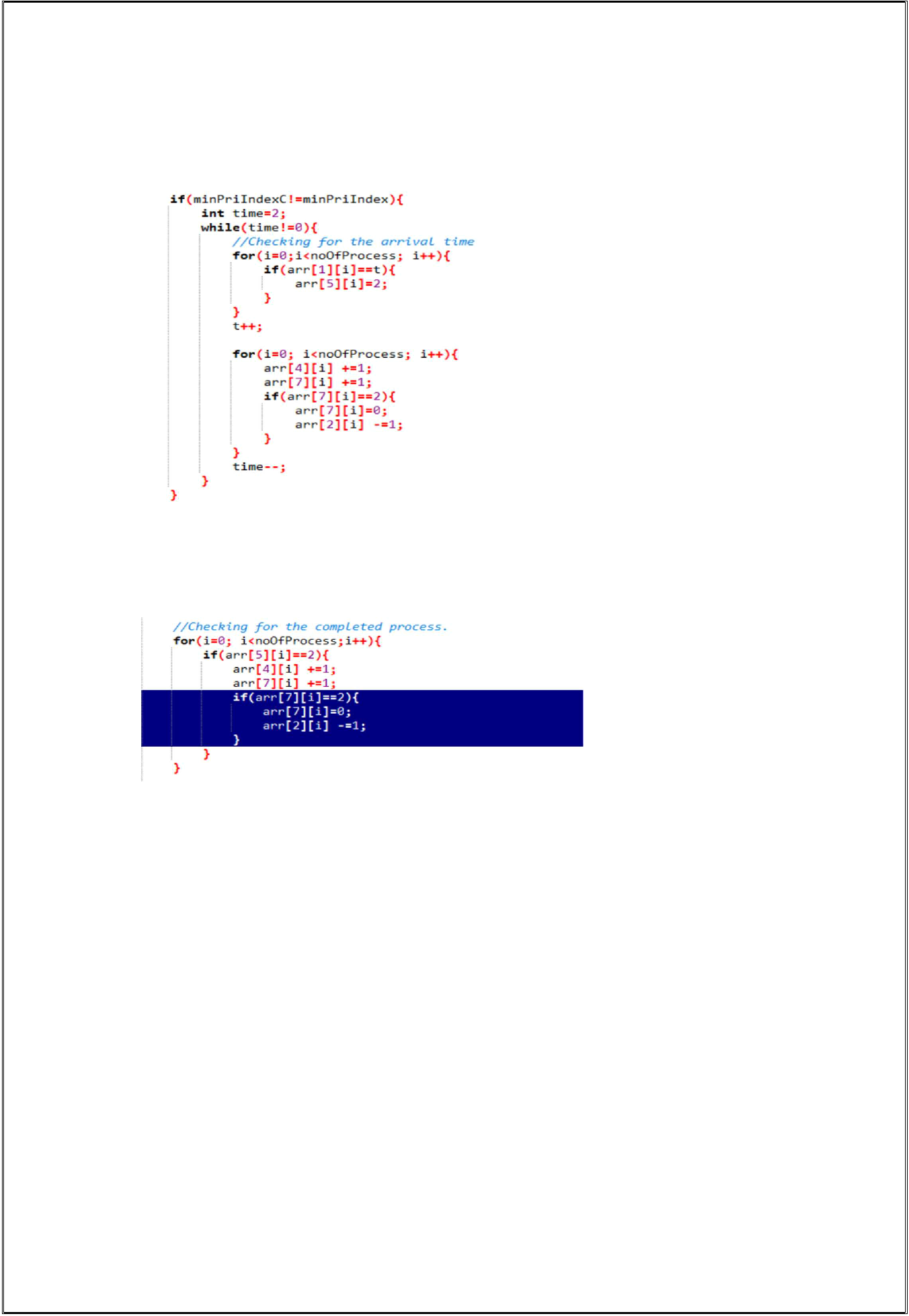
printf(" | ");

}

printf("\n\nTurn Around Time: %d second", t);

printf("\n\nAverage Waiting Time : %.2f second\n\n", totalWaitTime/(float)noOfProcess );

}

** 4. Constraints:**

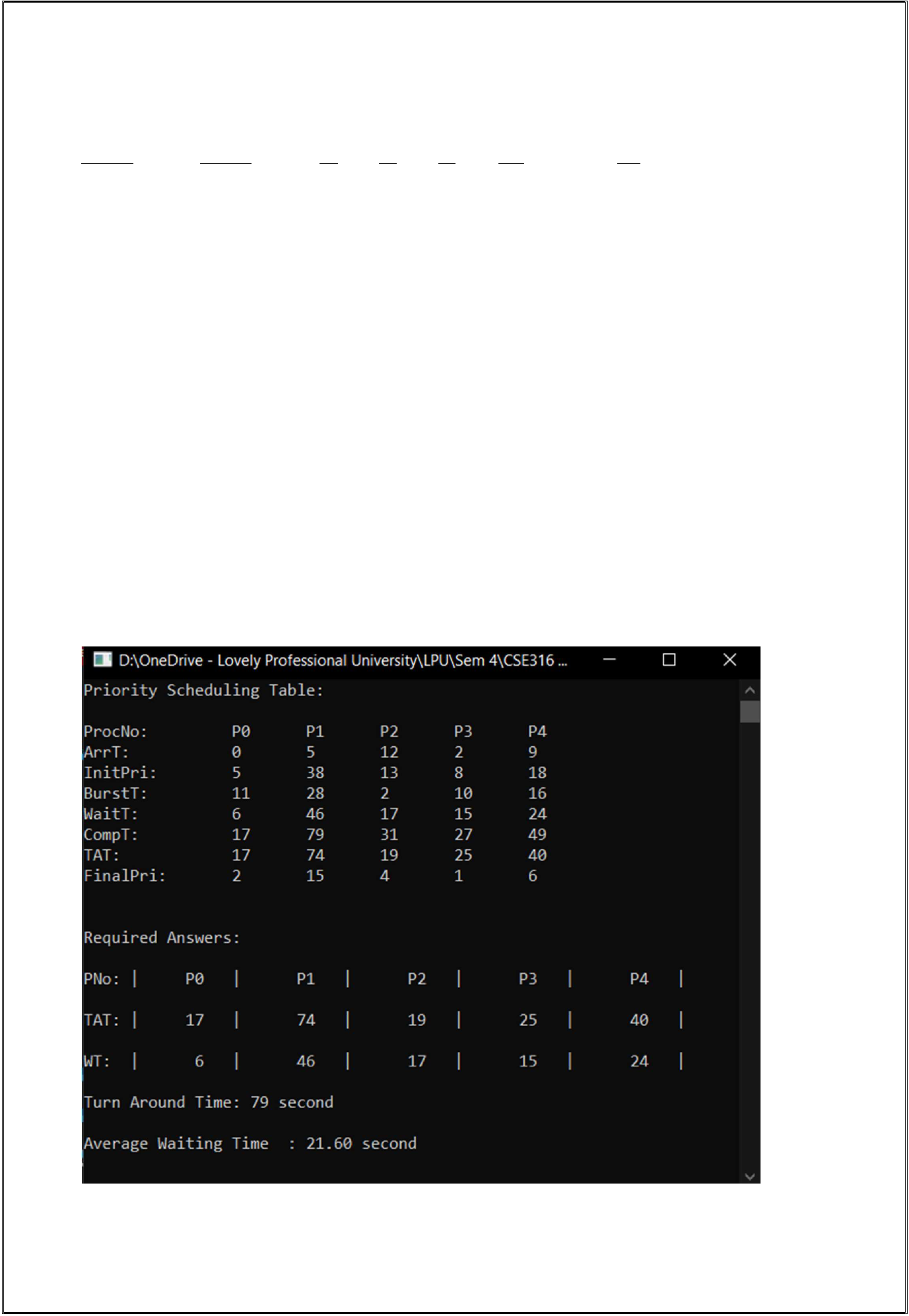
1. 2 seconds for context switching(dispatcher latency):

Here, when the processor is assigned from one process to another it takes 2 second and this is the dispatcher latency.

1. Priority of the process increase at rate of one per 2 time units of wait.

If the process waits for 2 second in ready state its priority increases one unit.

Here, in our case the priority number decreases by 1 because we are following Lower the number higher the priority approach.

1. **Description of additional algorithm used** No additional algorithms used
2. **Boundary Conditions:**
3. If the number of processes is less than 1 then the program may give unexpected behaviour.
4. The maximum number of process must be less than 100000.
5. **Test Cases:**
   1. **TestCase1 (Pass)**

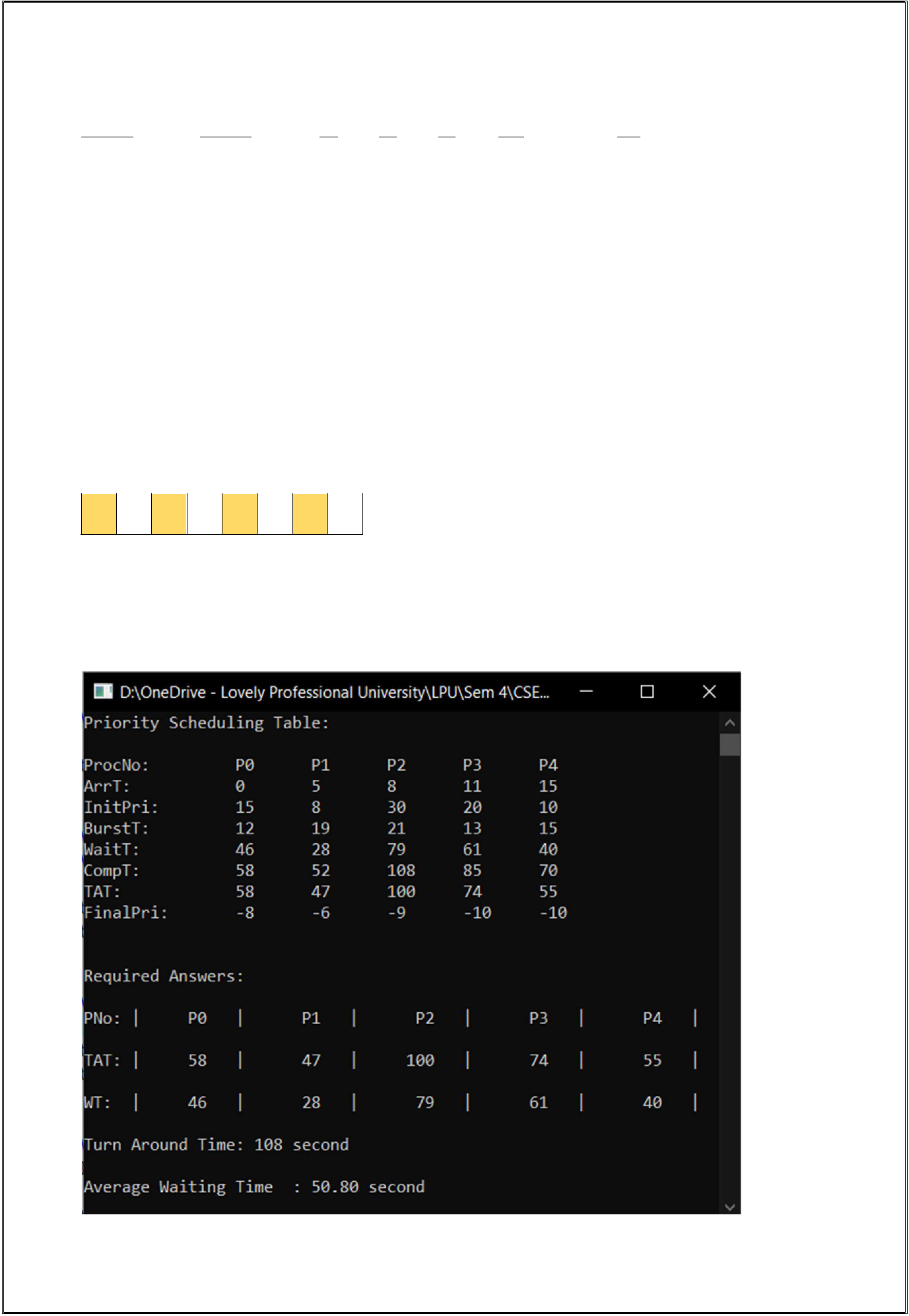
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Process** | | |  |  |  | **Priority** | | | | **AT** | |  | **BT** | | **CT** | | |  | **TAT***(=CT-AT)* | | | |  |  | **WT***(=TAT-BT)* | | | |  |  |  |
| **P0** |  |  |  |  |  | **5** |  |  |  | **0** |  |  | **11** | | **17** | | | 17-0= | | **17** | | |  |  | 17-11= | | **6** |  |  |  |  |
| **P1** |  |  |  |  |  | **38** |  |  |  | **5** |  |  | **28** | | **79** | | | 79-5= | | **74** | | |  |  | 74-28= **46** | | |  |  |  |  |
| **P2** |  |  |  |  |  | **13** |  |  |  | **12** |  |  | **2** | | **31** | | | 31-12= **19** | | | | |  |  | 19-2= |  | **17** |  |  |  |  |
| **P3** |  |  |  |  |  | **8** |  |  |  | **2** |  |  | **10** | | **27** | | | 27-2= | | **25** | | |  |  | 25-10= **15** | | |  |  |  |  |
| **P4** |  |  |  |  |  | **18** |  |  |  | **9** |  |  | **16** | | **49** | | | 49-9 = **40** | | | | |  |  | 40-16= **24** | | |  |  |  |  |
| Gantt Chart | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **P0** |  |  | DL | |  | **P3** |  | DL |  | **P0** |  | DL |  | **P3** |  |  | DL |  | **P2** |  |  | DL |  |  | **P4** |  | DL |  | **P1** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 10 | | 12 | | | 14 | | 16 | | 17 | |  | 19 | | 27 | |  | 29 | | 31 | |  | 33 | | 49 | | 51 | | 79 | |  |

**Turn Around Time = 79 second**

**Average Wait Time = 21.6 second**

Screenshot of the output for the above inputs:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **b)** | | **TestCase2 (Pass)** | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Process** | | | | |  |  |  | **Priority** | | | |  |  |  |  | **AT** | | |  |  | **BT** | |  | **CT** |  |  |  | **TAT***(=CT-AT)* | | | | | |  | **WT***(=TAT-BT)* | | | | |  |  |  |  |  |  |  |
| **P0** | |  |  |  |  |  |  | **15** |  |  |  |  |  |  |  | **0** |  |  |  |  | **12** |  | **58** | |  |  |  | 58-0= | | |  | **58** |  |  | 58-12= | | **46** | | |  |  |  |  |  |  |  |
| **P1** | |  |  |  |  |  |  | **8** |  |  |  |  |  |  |  | **5** |  |  |  |  | **19** |  | **52** | |  |  |  | 52-5= | | |  | **47** |  |  | 47-19= | | **28** | | |  |  |  |  |  |  |  |
| **P2** | |  |  |  |  |  |  | **30** |  |  |  |  |  |  |  | **8** |  |  |  |  | **21** |  | **108** | | |  |  | 108-8= **100** | | | | |  |  | 100-21= **79** | | | | |  |  |  |  |  |  |  |
| **P3** | |  |  |  |  |  |  | **20** |  |  |  |  |  |  |  | **11** | |  |  |  | **13** |  | **85** | |  |  |  | 85-11= | | | | **74** |  |  | 74-13= | | **61** | | |  |  |  |  |  |  |  |
| **P4** | |  |  |  |  |  |  | **10** |  |  |  |  |  |  |  | **15** | |  |  |  | **15** |  | **70** | |  |  |  | 70-55 = **55** | | | | |  |  | 55-15= | | **40** | | |  |  |  |  |  |  |  |
| Gantt Chart | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **P0** | |  | DL |  | **P1** | |  | DL |  | **P0** | |  | DL | **P1** | |  |  | DL |  |  | **P4** |  | DL | **P0** |  |  | DL |  |  | **P1** |  | DL |  | **P4** | DL | **P0** |  |  | DL |  |  | **P1** |  | DL |  | **P4** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  | 5 | 7 | | 21 | | | 23 | |  |  | 25 | | 27 | | 29 | |  | 31 | | 33 | | 35 | | 37 | |  | 39 | | 41 | | 43 | | 45 | 47 | | 49 | |  | 51 | | 52 | | 54 | | 55 |
|  |  |  |  | |  |  | |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | DL |  | **P0** | | DL |  | | **P4** | | DL |  |  | **P3** |  | DL |  |  | **P2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

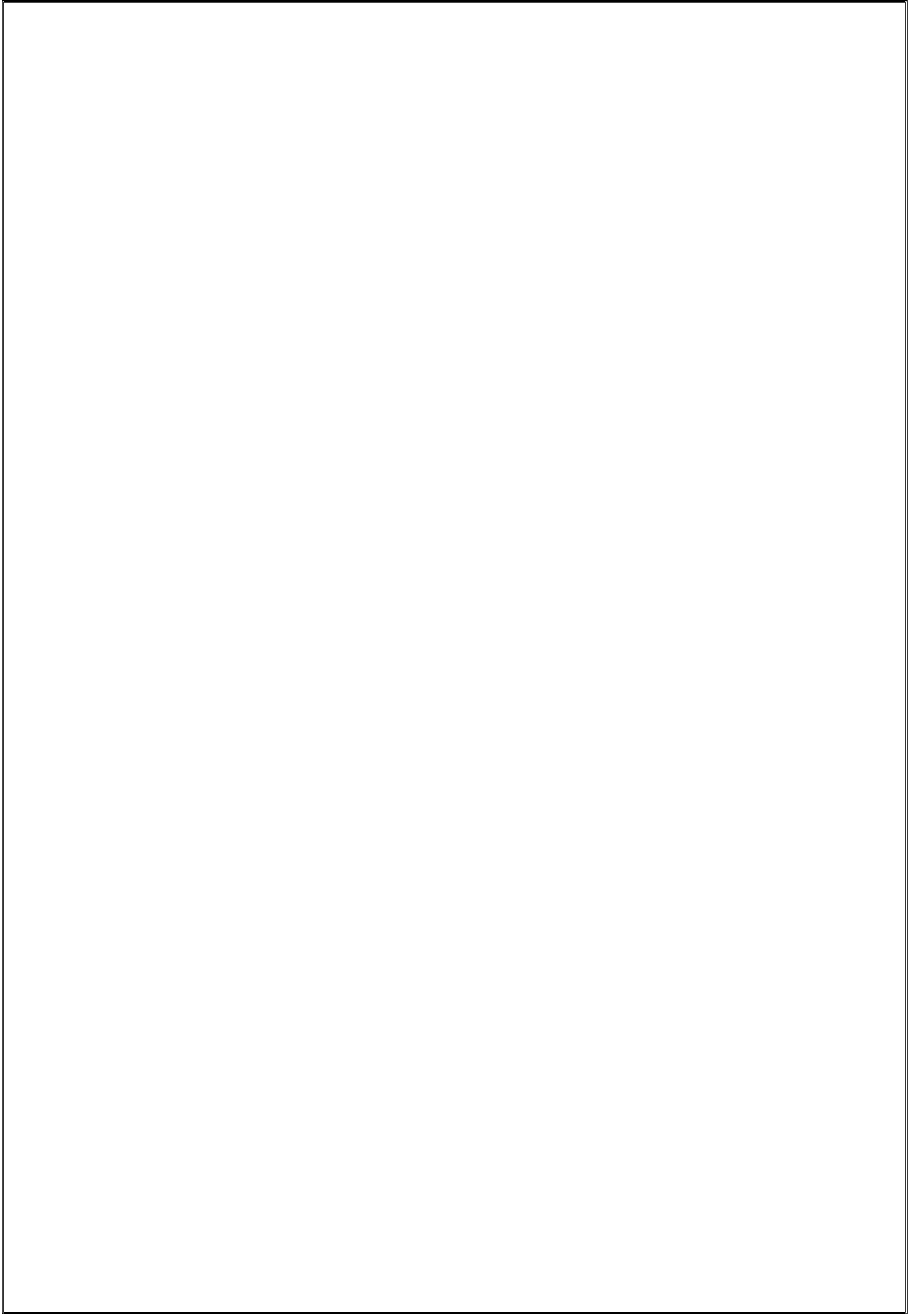


55 57 58 60 70 72 85 87 108

**Turn Around Time = 108 second**

**Average Wait Time = 50.8 second**

Screenshot of the output for the above inputs:

1. **GitHub Revisions:**

**Yes, I have made more than 5 revisions on the GitHub Repository.**

**Link:** <https://github.com/divyansh006/cse316>