Contents

[**USCS3P01:USCS303 – Operating System (OS) Practical\_01** 2](#_Toc77609428)

[**Practical – 01 : First Come First Serve (FCFS) Algorithm** 2](#_Toc77609429)

[**Practical Date : -** 16 / 07 /2021 2](#_Toc77609430)

[**Practical Aim :-** Implement FCFS Scheduling Algorithm in Java 2](#_Toc77609431)

[**Algorithm :-** 3](#_Toc77609432)

[**Gnatt Chart :-** 6](#_Toc77609433)

[**Implementation :-** 9](#_Toc77609434)

[**Input :-** 12](#_Toc77609435)

[**Output :-** 13](#_Toc77609436)

[**Sample Output – 01 :-** 14](#_Toc77609437)

[**Sample Output – 02 :-** 15](#_Toc77609438)

[**Sample Output – 03 :-** 16](#_Toc77609439)

[**Sample Output 04 :-** 17](#_Toc77609440)

# **USCS3P01:USCS303 – Operating System (OS) Practical\_01**

## **Practical – 01 : First Come First Serve (FCFS) Algorithm**

### **Practical Date : -** 16/ 07 /2021

### **Practical Aim :-** Implement FCFS Scheduling Algorithm in Java

CPU Scheduling algorithms are used for scheduling different processes present in the ready queue with available resources(CPU cores) in an optimal way so that each and every process get executed by CPU.

Scheduling algorithms are broadly classified into two main types name Preemptive and Non-preemptive. First Come First Service (FCFS) is a Non-preemptive Scheduling algorithm where each process is executed according to its arrival time.

First Come First Serve (FCFS) is also know as First In First Out (FIFO) scheduling algorithm is the easiest and simplest CPU scheduling algorithm where the process which arrives first in the ready queue is executed first by the CPU.

New process is executed only when the current process is executed fully by the CPU

### **Algorithm :-**

**In Algorithm we need to know :**

* Finish Time = Previous Process Finish Time + Burst Time
* Turn Around Time = Finish Time – Arrival Time
* Waiting Time = Turn Around Time – Burst time

**FCFS Algorithm** :

**Step 1** : Input the number of processes required to be scheduled using FCFS , burst time for each process and its arrival time .

**Step 2** : Using enhanced bubble sort technique , sort the all given processes in ascending order according to arrival time in a ready queue.

**Step 3** : Calculate the Finish time , Turn Around Time and Waiting Time for each process which in turn help to calculate Average Waiting Time and Average Turn Around Time required by CPU to schedule given set of process using FCFS.

**Step 3.1** : For i = 0, Finish Time T0 = Arrival Time T0 + Burst Time T0

**Step 3.2** : For i >= 1, Finish Time Ti = Burst Time Ti + Finish Time Ti-1

**Step 3.3** : For i = 0, Turn Around Time T0 = Finish Time T0  - Arrival Time T0

**Step 3.4** : For i = 1, Turn Around Time Ti = Finish Time Ti - Arrival Time Ti

**Step 3.5** : For i = 0, Waiting Time T0 = Turn Around Time T0 - Burst Time T0

**Step 3.6** : For i = 1, Waiting Time Ti = Turn Around Time Ti  - Burst Time Ti-1

**Step 4** : Process with less arrival time comes first and gets scheduled first by the CPU.

**Step 5** : Calculate the Average Waiting Time and Average Turn Around Time.

**Step 6** : Stop.

**Solved Example :-**

**Example 1** :- Consider the following example containing five processes arrive at same time.

|  |  |
| --- | --- |
| Process ID | Burst Time |
| P0 | 6 |
| P1 | 3 |
| P2 | 8 |
| P3 | 3 |
| P4 | 4 |

**Step 1 :** Processes get executed according to their arrival time.

**Step 2 :** Following Shows the scheduling and execution of processes.

**Step 2.1 :** At start P0 arrives and get executed for 6 (i.e; 0 – 6 ) seconds.

|  |  |
| --- | --- |
| **Process ID** | **Burst Time** |
| P0 | 6 |
| P1 | 3 |
| P2 | 8 |
| P3 | 3 |
| P4 | 4 |

**Step 2.2 :** P1 arrives after completion of P0 , P1 is executed for 3 (i.e; 6 – 9 )

Seconds.

|  |  |  |
| --- | --- | --- |
| System Time | : | 0 |
| Process Scheduled | : | P0 |
| Waiting Time | : | 0 + 0 = 0 |
| Turn Around Time | : | 0 + 6 = 6 |

**Step 2.2:** P1 arrives after completion of P0,P1 is executed for 3 (i.e; 6 – 9) seconds.

|  |  |  |
| --- | --- | --- |
| System Time | : | 6 |
| Process Scheduled | : | P0, P1 |
| Waiting Time | : | 6 + 0 = 6 |
| Turn Around Time | : | 3 + 6 = 9 |

**Step 2.3:** P2 arrives after complete execution of process P1, for 8 (i.e; 9 – 17) seconds

|  |  |  |
| --- | --- | --- |
| System Time | : | 17 |
| Process Scheduled | : | P0, P1, P2, P3 |
| Waiting Time | : | 8 + 9 = 17 |
| Turn Around Time | : | 3 + 17 = 20 |

**Step 2.4:** P3 arrives and gets executed for 3 (i.e; 17 – 20) seconds.

|  |  |  |
| --- | --- | --- |
| System TIme | : | 17 |
| Process Scheduled | : | P0, P1, P2, P3 |
| Waiting Time | : | 8 + 9 = 17 |
| Turn Around Time | : | 3 + 17 = 20 |

**Step 2.5:** Similarly, P4 arrives and gets executed for 4 (i.e; 20 – 24) seconds.

|  |  |  |
| --- | --- | --- |
| System Time | : | 20 |
| Process Scheduled | : | P0, P1, P2, P3, P4 |
| Waiting Time | : | 3 + 17 = 20 |
| Turn Around Time | : | 4 + 20 = 24 |

**Step 3:** Calculate Average Waiting and Average Turn Around Time

**Average Waiting Time =**(0 + 6 + 9 + 17 + 20) / 5 = 52 / 5 = **10.4**

**Average Turn Around Time =** (6 + 9 + 17 + 20 + 24) / 5 = 76 / 5 = **15.2**

**Step 4:** After scheduling of all provided processes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process ID** | **Burst Time** | **Arrival Time** | **Finish Time** | **Turn Around Time** | **Waiting TIme** |
| P0 | 6 | 0 | (-+6=)6 | (6-0=)6 | (6-6=)0 |
| P1 | 3 | 0 | (6+3=)9 | (9-0=)9 | (9-3=)6 |
| P2 | 8 | 0 | (9+8=)17 | (17-0=)17 | (17-8=)9 |
| P3 | 3 | 0 | (17+3=)20 | (20-0=)20 | (20-3=)17 |
| P4 | 4 | 0 | (20+4=)24 | (24-0)=24 | (24-4=)20 |
| **Average** |  |  |  | 15.200000 | 10.400000 |

**Step 5:** Stop

### **Gnatt Chart :-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P0** | **P1** | **P2** | **P3** | **P4** |  |
| 0 | 6 | 9 | 17 | 20 | 24 |

**Example 2** :- Consider the Following example containing five processes with varied arrival time.

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Burst Time** | **Arrival Time** |
| P0 | 6 | 2 |
| P1 | 3 | 5 |
| P2 | 8 | 1 |
| P3 | 3 | 0 |
| P4 | 4 | 4 |

**Solution:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process**  **ID** | **Burst**  **Time** | **Arival**  **Time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time)** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| P3 | 3 | 0 | 3 | 3 | 0 |
| P2 | 8 | 1 | 11 | 10 | 2 |
| P0 | 6 | 2 | 17 | 15 | 9 |
| P4 | 4 | 4 | 21 | 17 | 13 |
| P1 | 3 | 5 | 24 | 19 | 16 |
| Average |  |  |  | 12.800000 | 8.000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P0** | **P1** | **P2** | **P3** | **P4** |  |
| 0 | 3 | 11 | 17 | 21 | 24 |

**Example:03**

Consider the following example containing five processes arrive at same time.

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Burst Time** | **Arrival Time** |
| P0 | 2 | 2 |
| P1 | 1 | 5 |
| P2 | 6 | 1 |

**Solution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process**  **ID** | **Burst**  **Time** | **Arival**  **Time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time)** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| **P0** | **2** | **0** | **2** | **2** | **0** |
| **P1** | **1** | **0** | **3** | **3** | **2** |
| **P2** | **6** | **0** | **9** | **9** | **3** |
| **Average** |  |  |  | **4.66667** | **1.66667** |

|  |  |  |  |
| --- | --- | --- | --- |
| **P0** | **P1** | **P2** |  |
| 0 | 2 | 3 | 9 |

**Example:04**

Consider the following example containing five processes with varied arrival time.

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Burst Time** | **Arrival Time** |
| P0 | 4 | 3 |
| P1 | 3 | 5 |
| P2 | 2 | 0 |
| P3 | 1 | 5 |
| P4 | 3 | 4 |

**Solution:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process**  **ID** | **Burst**  **Time** | **Arival**  **Time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time)** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| P2 | 2 | 0 | 2 | 2 | 0 |
| P0 | 4 | 3 | 6 | 3 | -1 |
| P4 | 3 | 4 | 9 | 5 | 2 |
| P1 | 3 | 5 | 12 | 7 | 4 |
| P3 | 1 | 5 | 13 | 8 | 7 |
| **Average** |  |  |  | **5.000000** | **2.400000** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **P0** | **P1** | **P2** | **P3** | **P4** |  |
| 0 | 2 | 6 | 9 | 12 | 13 |

### **Implementation :-**

import java.util.Scanner;

public class P1\_FCFS\_YP

{

int burstTime[];

int arrivalTime[];

String[] processId;

int numberOfProcess;

void getProcessData(Scanner input){

System.out.println("enter the number of process for Scheduling:");

int inputNumberOfProcess=input.nextInt();

numberOfProcess=inputNumberOfProcess;

burstTime=new int[numberOfProcess];

arrivalTime=new int[numberOfProcess];

processId=new String[numberOfProcess];

String st="p";

for(int i=0;i < numberOfProcess;i++){

processId[i]=st.concat(Integer.toString(i));

System.out.print("enter the burst time for process-"+(i)+":");

burstTime[i]=input.nextInt();

System.out.println("enter the arrival time for process-"+(i)+":");

arrivalTime[i]=input.nextInt();

}

}

void sortAccordingArrivalTime(int[] at,int[] bt,String[] pid){

boolean swapped;

int temp;

String stemp;

for (int i=0;i<numberOfProcess;i++){

swapped=false;

for (int j = 0;j<numberOfProcess-i-1;j++){

if(at[j]>at[j+1]){

temp=at[j];

at[j]=at[j+1];

at[j+1]=temp;

temp=bt[j];

bt[j]=bt[j+1];

bt[j+1]=temp;

stemp=pid[j];

pid[j]=pid[j+1];

pid[j+1]=stemp;

swapped=true;

}

}

if(swapped==false){

break;

}

}

}

void firstComeFirstServeAlgorithm(){

int finishTime[]=new int[numberOfProcess];

int bt[]=burstTime.clone();

int at[]=arrivalTime.clone();

String pid[]=processId.clone();

int waitingTime[]=new int[numberOfProcess];

int turnAroundTime[]=new int[numberOfProcess];

sortAccordingArrivalTime(at,bt,pid);

finishTime[0]=at[0]+bt[0];

turnAroundTime[0]=finishTime[0]-at[0];

waitingTime[0]=turnAroundTime[0]-bt[0];

for(int i=1;i<numberOfProcess;i++){

finishTime[i]=bt[i]+finishTime[i-1];

turnAroundTime[i]=finishTime[i]-at[i];

waitingTime[i]=turnAroundTime[i]-bt[i];

}

float sum=0;

for(int n:waitingTime){

sum+=n;

}

float averageWaitingTime=sum/numberOfProcess;

sum=0;

for(int n:turnAroundTime){

sum+=n;

}

float averageTurnAroundTime=sum/numberOfProcess;

System.out.println("FCFS Scheduling algorithm :");

System.out.format("%20s%20s%20s%20s%20s%20s\n","ProcessId","BurstTime"

,"ArrivalTime","FinishTime","TurnAroundTime","WatingTime");

for(int i=0;i<numberOfProcess;i++){

System.out.format("%20s%20d%20d%20d%20d%20d\n",pid[i],bt[i],at[i]

,finishTime[i],turnAroundTime[i],waitingTime[i]);

}

System.out.format("%80s%20f%20f\n", "Average",averageTurnAroundTime,averageWaitingTime);

}

public static void main(String[] args){

Scanner input=new Scanner(System.in);

P1\_FCFS\_YP obj=new P1\_FCFS\_YP();

obj.getProcessData(input);

obj.firstComeFirstServeAlgorithm();

}

}

### **Input :-**

Enter the number of process for scheduling:5

Enter the burst time for Process0:

Enter the arrival time for Process0:

Enter the burst time for Process1:

Enter the arrival time for Process1:

Enter the burst time for Process2:

Enter the arrival time for Process2:

Enter the burst time for Process3:

Enter the arrival time for Process3:

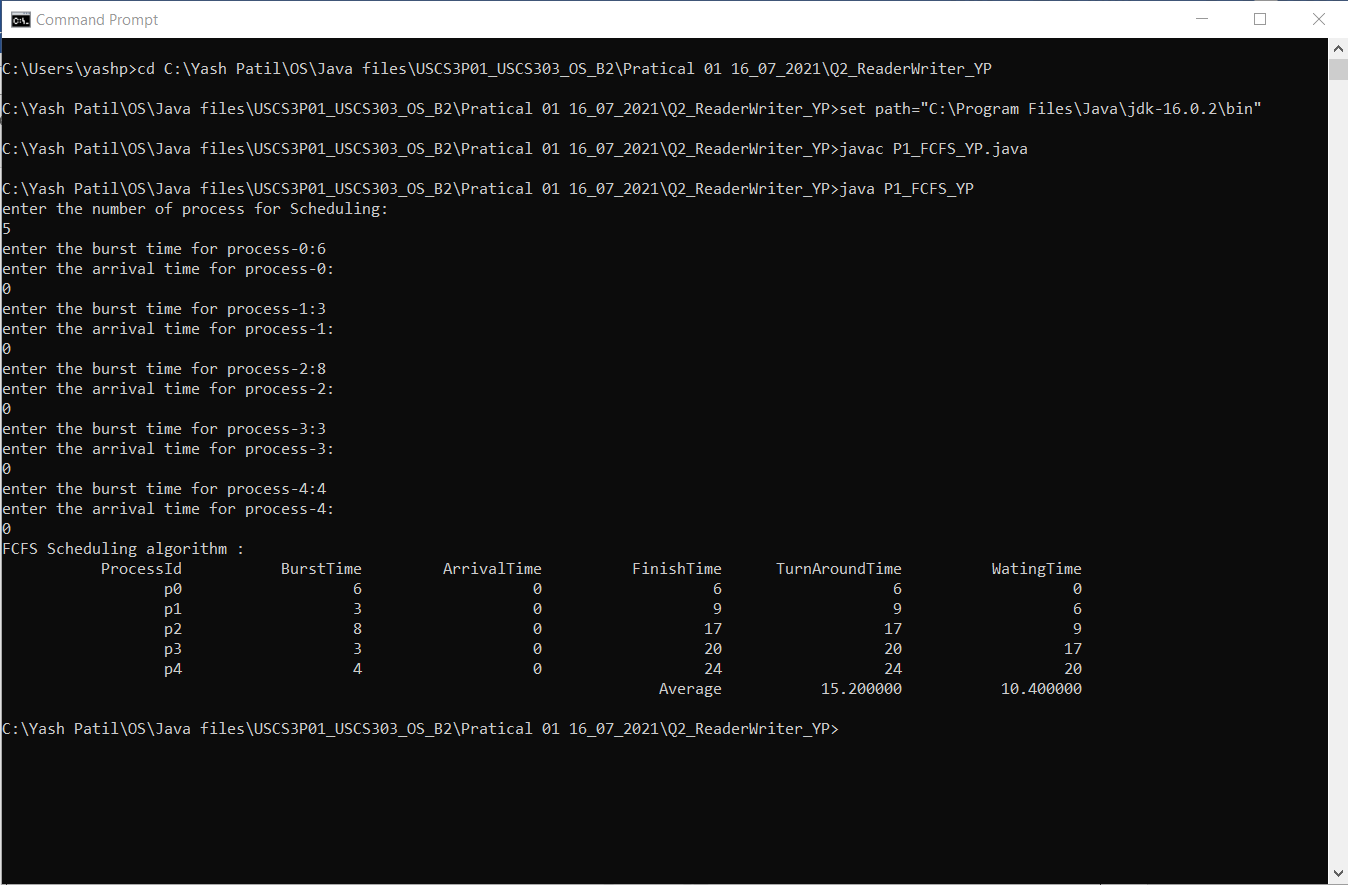
Enter the burst time for Process4:

Enter the arrival time for Process4:

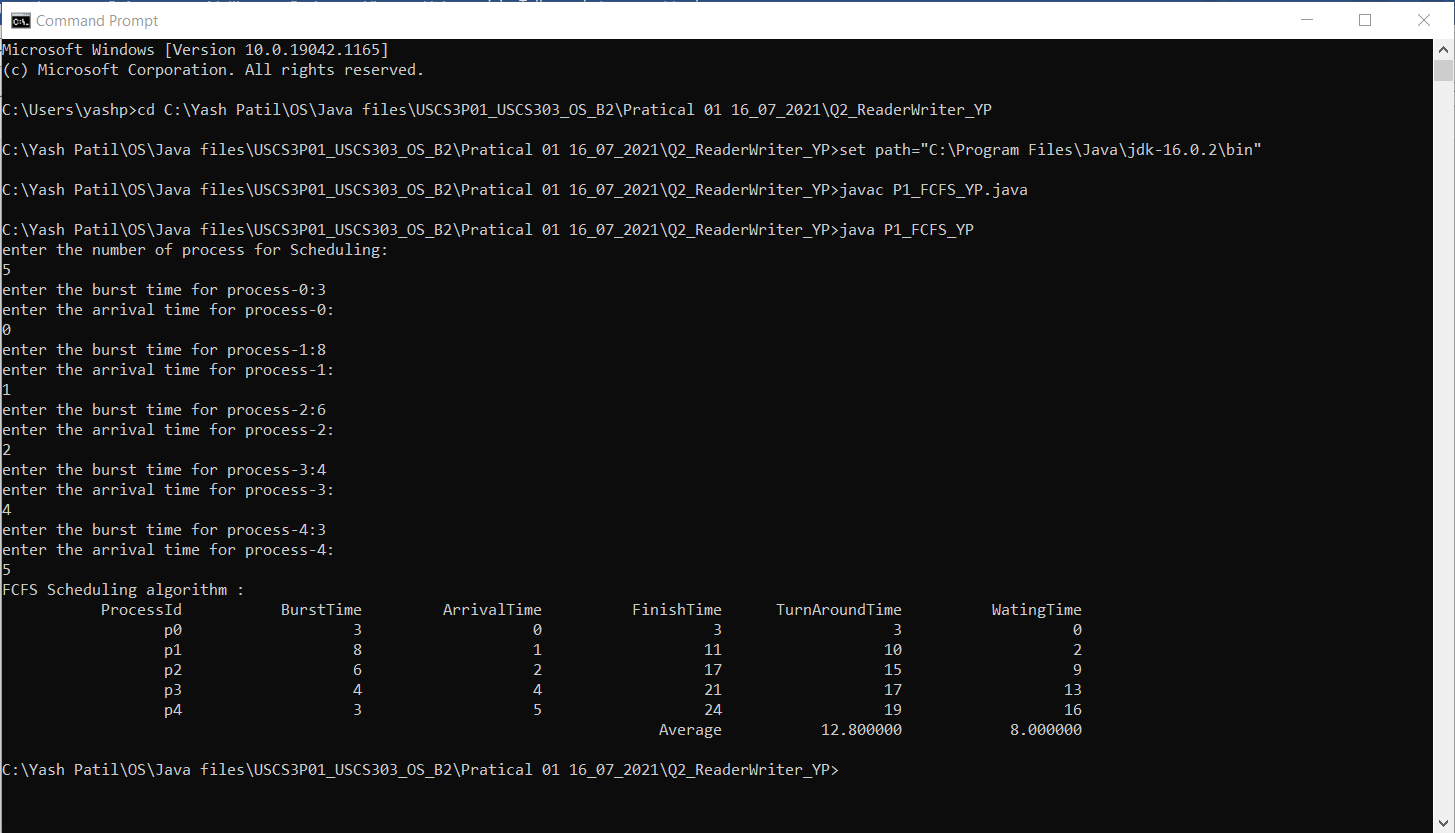
### **Output :-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process**  **ID** | **Burst**  **Time** | **Arival**  **Time** | **Finish Time**  **(Prev.Finish**  **Time + Burst**  **Time)** | **Turn Around**  **Time**  **(Finish Time- Arrival Time)** | **Waiting Time**  **(Turn Around**  **Time-Burst**  **Time)** |
| P3 | 3 | 0 | 3 | 3 | 0 |
| P2 | 8 | 1 | 11 | 10 | 2 |
| P0 | 6 | 2 | 17 | 15 | 9 |
| P4 | 4 | 4 | 21 | 17 | 13 |
| P1 | 3 | 5 | 24 | 19 | 16 |
| Average |  |  |  | 12.800000 | 8.000000 |

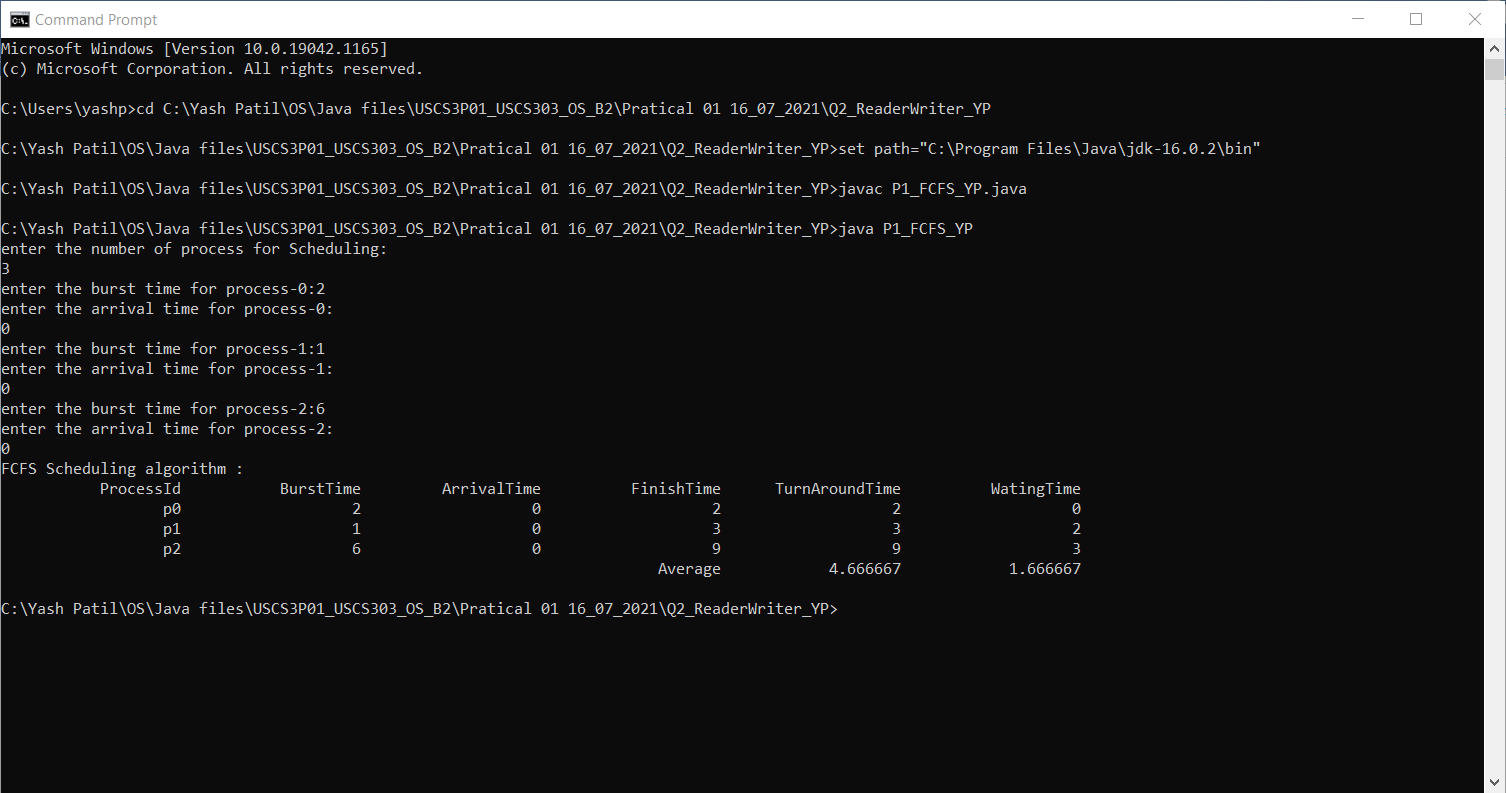
### **Sample Output – 01 :-**



### **Sample Output – 02 :-**



### **Sample Output – 03 :-**



### **Sample Output 04 :-**

