Lab # 6

Process Scheduling (i) First Come First Serve (FCFS) & (ii) Shortest Job First (SJF)

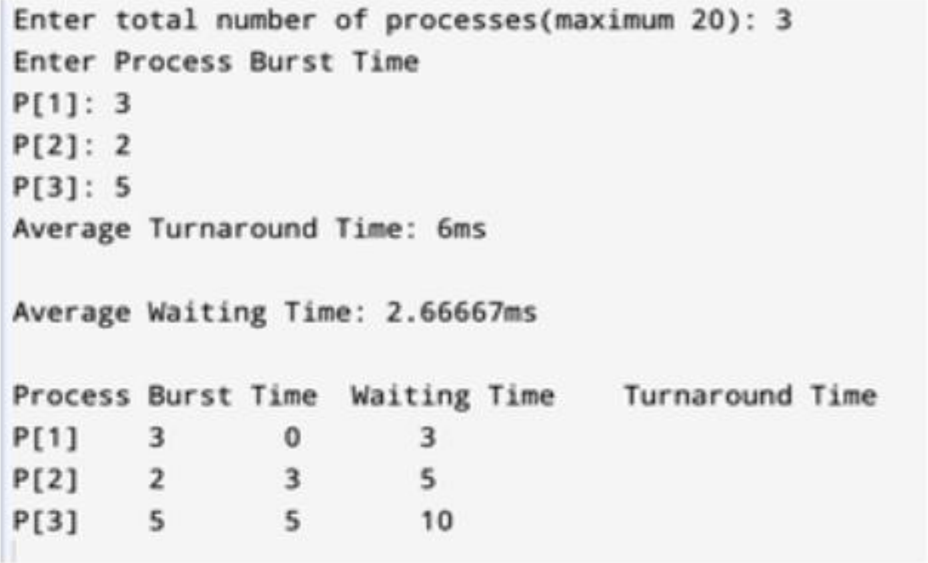
Write shell scripting code for FCFS scheduling and also calculate average waiting time.

A. First Come First Serve Scheduling Algorithm (FCFS) :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

class Program {
        static void Main(string[] args)
        {
                int n;
                int[] burst_time = new int[20];
                int[] waiting_time = new int[20];
                int[] turnaround_time = new int[20];
                float avg_waiting_time = 0;
                float avg_turnaround_time = 0;

                Console.Write(
                        "Enter total number of processes(maximum 20): ");
                n = Convert.ToInt32(Console.ReadLine());

                Console.WriteLine("\nEnter Process Burst Time");
                for (int i = 0; i < n; i++) {
                        Console.Write("P[" + (i + 1) + "]: ");
                        burst_time[i]
                                = Convert.ToInt32(Console.ReadLine());
                }

                waiting_time[0]
                        = 0; // Waiting time for first process is 0

                // Calculating waiting time
                for (int i = 1; i < n; i++) {
                        waiting_time[i] = 0;
                        for (int j = 0; j < i; j++) {
                                waiting_time[i] += burst_time[j];
                        }
                }

                // Calculating turnaround time by adding burst_time
                // and waiting_time
                for (int i = 0; i < n; i++) {
```

```
    turnaround_time[i]
        = burst_time[i] + waiting_time[i];
    avg_turnaround_time += turnaround_time[i];
}

avg_turnaround_time /= n;
Console.WriteLine("\nAverage Turnaround Time: "
                + avg_turnaround_time + "ms\n");

// Calculating average waiting time
for (int i = 0; i < n; i++) {
    avg_waiting_time += waiting_time[i];
}

avg_waiting_time /= n;
Console.WriteLine("\nAverage Waiting Time: "
                + avg_waiting_time + "ms\n\n");

Console.WriteLine(
    "Process\tBurst Time\tWaiting Time\tTurnaround Time");
for (int i = 0; i < n; i++) {
    Console.WriteLine("P[" + (i + 1) + "]\t"
                    + burst_time[i] + "\t\t"
                    + waiting_time[i] + "\t\t"
                    + turnaround_time[i]);
} } }
```

Output:

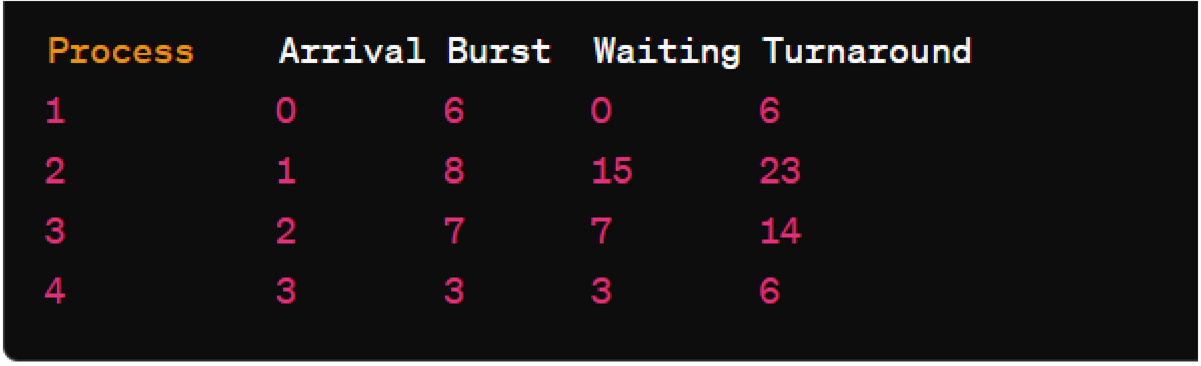


```
Enter total number of processes(maximum 20): 3
Enter Process Burst Time
P[1]: 3
P[2]: 2
P[3]: 5
Average Turnaround Time: 6ms

Average Waiting Time: 2.66667ms

Process Burst Time  Waiting Time    Turnaround Time
P[1]    3       0       3
P[2]    2       3       5
P[3]    5       5       10
```

B. Shortest Job First Scheduling Algorithm (SJF)

```
using System;
using System.Collections.Generic;

public class Process
{
    public int Id { get; set; }
    public int ArrivalTime { get; set; }
    public int BurstTime { get; set; }
    public int WaitingTime { get; set; }
    public int TurnaroundTime { get; set; }

    public Process(int id, int arrivalTime, int burstTime)
    {
        Id = id;
        ArrivalTime = arrivalTime;
        BurstTime = burstTime;
        WaitingTime = 0;
        TurnaroundTime = 0;
    }
}

public class ShortestJobFirst
{
    public List<Process> Processes { get; set; }

    public ShortestJobFirst(List<Process> processes)
    {
        Processes = processes;
    }

    public void Schedule()
    {
        // Sort processes based on arrival time and burst time
        Processes.Sort((a, b) => {
            int compareArrival = a.ArrivalTime.CompareTo(b.ArrivalTime);
            if (compareArrival == 0)
            {
                return a.BurstTime.CompareTo(b.BurstTime);
            }
            return compareArrival;
        });

        int currentTime = 0;
        int completedProcesses = 0;
        int n = Processes.Count;
        while (completedProcesses < n)
        {
            // Find the next shortest job that has arrived
            Process shortestProcess = null;
            for (int i = 0; i < n; i++)
            {
                Process proc = Processes[i];
                if (proc.ArrivalTime <= currentTime && proc.BurstTime > 0)
                {
                    if (shortestProcess == null || proc.BurstTime <
shortestProcess.BurstTime)
                    {
                        shortestProcess = proc;
                    }
                }
            }

            if (shortestProcess != null)
            {
                // Process the shortest process found
                currentTime += shortestProcess.BurstTime;
                shortestProcess.TurnaroundTime = currentTime -
shortestProcess.ArrivalTime;
                shortestProcess.WaitingTime = shortestProcess.TurnaroundTime -
shortestProcess.BurstTime;

                // Mark the process as completed
                shortestProcess.BurstTime = 0;
                completedProcesses++;
            }
            else
            {
                // If no process found to execute, just increment the time
                currentTime++;
            }
        }
    }

    public void DisplayResults()
    {
        Console.WriteLine("Process\tArrival\tBurst\tWaiting\tTurnaround");
        foreach (var proc in Processes)
        {

Console.WriteLine($"{proc.Id}\t{proc.ArrivalTime}\t{proc.BurstTime}\t{proc.Wai
tingTime}\t{proc.TurnaroundTime}");
        }
    }
}

public class Program
{
    public static void Main()
    {
        // Define the list of processes
        List<Process> processes = new List<Process>
        {
            new Process(1, 0, 6),
            new Process(2, 1, 8),
            new Process(3, 2, 7),
            new Process(4, 3, 3)
        };

        // Create a SJF scheduler
        ShortestJobFirst sjf = new ShortestJobFirst(processes);

        // Schedule the processes
        sjf.Schedule();

        // Display the results
        sjf.DisplayResults();
    }
}
```

Output



```
Process     Arrival Burst  Waiting Turnaround
1           0       6      0       6
2           1       8      15      23
3           2       7      7       14
4           3       3      3       6
```

Lab Tasks

1. For below processes, write FCFS and SJF processes to calculate waiting time for each process and average waiting time.

Process	Arrival Time	Execute Time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

2. For below processes, write FCFS and SJF processes to calculate waiting time for each process and average waiting time.

Process	Arrival Time	Execute Time
P0	2	6
P1	5	2
P2	1	8
P3	0	3
P4	4	4