**ASSIGNMENT – 6**

**1. WAP to implement FCFS scheduling algorithm (in programming language of your choice). Consider process name, burst time, waiting time and turnaround time as the important parameters.**

**ALGORITHM:**

1. Completion time of each of the processes is calculated first. A variable named currentTime is initialized with its value as 0. Completion time is then calculated with the help of the formula current time + burst time. And current time is then changed as current time + burst time.

for (i = 0; i < n; i++)

{

currentTime += burstTime[i];

completionTime[i] = currentTime;

}

1. The value of turnaround time is equal to completion time as we assumed that all the processes are arriving at time = 0.
2. Total turnaround time is also calculated simultaneously as average turnaround time.
3. Then we calculate waiting time.

Waiting time = turnaround time – burst time

1. Finally, we calculate average waiting time and average turnaround time.

**Implementation of above logic -**

*#include* <stdio.h>

*#include* <stdlib.h>

int main()

{

    int i, n;

    printf("Enter number of processes : ");

    scanf("%d", &n);

    int \*burstTime = (int\*)malloc(n \* sizeof(int));

    int \*waitingTime = (int\*)malloc(n \* sizeof(int));

    int \*completionTime = (int\*)malloc(n \* sizeof(int));

    int \*turnaroundTime = (int\*)malloc(n \* sizeof(int));

    int currentTime = 0;

    float avgWT = 0, avgTT = 0;

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

*for* (i = 0; i < n; i++) {

        int temp;

        scanf("%d", &temp);

        burstTime[i] = temp;

    }

*for* (i = 0; i < n; i++) {

        currentTime += burstTime[i];

        completionTime[i] = currentTime;

    }

*for* (i = 0; i < n; i++) {

        turnaroundTime[i] = completionTime[i] - 0;

        avgTT += turnaroundTime[i];

    }

*for* (i = 0; i < n; i++) {

        waitingTime[i] = turnaroundTime[i] - burstTime[i];

        avgWT += waitingTime[i];

    }

    printf("-------------------------------------------------------------\n\t\tFirst Come First Serve\n-------------------------------------------------------------\n");

    printf("    Process No. \tBurstTime \tWaiting Time \tTurn Around Time\n");

*for*(i = 0; i < n; i++) {

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

    }

    avgTT = avgTT / n;

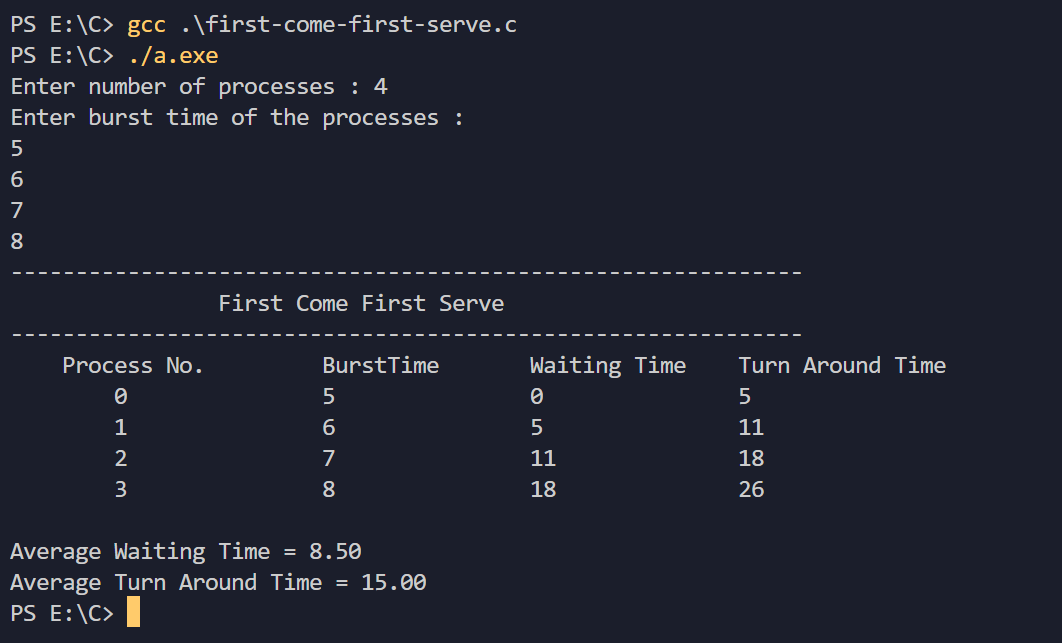
    avgWT = avgWT / n;

    printf("\nAverage Waiting Time = %.2f \nAverage Turn Around Time = %.2f\n", avgWT, avgTT);

*return* 0;

}

**OUTPUT:**



**ASSIGNMENT – 7**

**1. WAP to implement non-pre-emptive SJF scheduling algorithm (in programming language of your choice). Consider process name, burst time, waiting time and turnaround time as the important parameters. Consider arrival time as same for all processes.**

**ALGORITHM:**

1. First, we sort the array of burst time in ascending order.
2. As all the processes arrives at time = 0 and the algorithm is non-pre-emptive, so the processes will be assigned, with respect to this sorted array.
3. Here, we use map data structure in C++, we mapped the burst time of each process with their process number.
4. We run a for loop executing all the processes one by one. We keep a track of current time and assign values of completion time, turnaround time, waiting time.

Completion time = current time + burst time of current process

Turnaround time = Completion time

Waiting time = Turnaround time – burst time of the process

= Current time.

1. Finally, we get average turnaround time and average waiting time by dividing total by n, where n = number of processes.

**Implementation of the above logic -**

*#include* <bits/stdc++.h>

using namespace std;

int main() {

    cout << "Enter number of processes: ";

    int n;

    cin >> n;

    map<int, int> processes;

    cout << "Enter burst time of each process: " << endl;

*for* (int i = 0; i < n; i++) {

        int temp;

        cin >> temp;

        processes.insert(make\_pair(temp, i));

    }

    int waitingTime[n], turnAroundTime[n], completionTime[n];

    int avgWT = 0, avgTAT = 0, currentTime = 0;

*for* (auto value: processes) {

        int time = value.first;

        int processNumber = value.second;

        completionTime[processNumber] = currentTime + time;

        waitingTime[processNumber] = currentTime;

        turnAroundTime[processNumber] = completionTime[processNumber];

        avgWT += waitingTime[processNumber];

        avgTAT += turnAroundTime[processNumber];

        currentTime += time;

    }

    cout << "Process\t\tBurst Time\t\tCompletion Time\t\t\tTurn Around Time\t\tWaiting Time" << endl;

*for* (auto value: processes) {

        int i = value.second;

        cout << "   P" << i + 1 << "\t\t    " << value.first << "\t\t\t     " << completionTime[i] << "\t\t\t\t    " << turnAroundTime[i] << "\t\t\t\t    " << waitingTime[i] << endl;

    }

    cout << "Average Waiting Time: " << avgWT << endl;

    cout << "Average Turn Around Time: " << avgTAT << endl;

*return* 0;

}

**OUTPUT:**

