**Experiment-3: Design a CPU scheduling program with C using First Come First Served technique with the following considerations.**

**a. All processes are activated at time 0.**

**b. Assume that no process waits on I/O devices**

Aim:

To simulate the CPU scheduling using the First Come First Served (FCFS) technique.

Procedure:

1. Input the arrival time and burst time for each process.
2. Calculate the completion time, turnaround time, and waiting time for each process.
3. Display the scheduling table.

C Program:

#include <stdio.h>

struct Process {

int id;

int arrival\_time;

int burst\_time;

int completion\_time;

int waiting\_time;

int turnaround\_time;

};

int main() {

int n;

printf("Enter number of processes: ");

scanf("%d", &n);

struct Process processes[n];

int total\_waiting\_time = 0, total\_turnaround\_time = 0;

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

processes[i].id = i + 1;

printf("Enter arrival time and burst time for process %d: ", i + 1);

scanf("%d %d", &processes[i].arrival\_time, &processes[i].burst\_time);

}

processes[0].completion\_time = processes[0].arrival\_time + processes[0].burst\_time;

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

processes[i].completion\_time = processes[i-1].completion\_time + processes[i].burst\_time;

}

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

processes[i].turnaround\_time = processes[i].completion\_time - processes[i].arrival\_time;

processes[i].waiting\_time = processes[i].turnaround\_time - processes[i].burst\_time;

total\_waiting\_time += processes[i].waiting\_time;

total\_turnaround\_time += processes[i].turnaround\_time;

}

printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].id, processes[i].arrival\_time, processes[i].burst\_time, processes[i].waiting\_time, processes[i].turnaround\_time);

}

printf("\nAverage Waiting Time: %.2f\n", (float)total\_waiting\_time / n);

printf("Average Turnaround Time: %.2f\n", (float)total\_turnaround\_time / n);

return 0;

}

Output:

