**Experiment:4-Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next**

Aim:

To implement the Shortest Job Next (SJN) CPU Scheduling algorithm.

Procedure:

1. Sort processes based on their burst time in ascending order.
2. Calculate waiting time, turnaround time, and display the scheduling order.

C Program:

#include <stdio.h>

#include <stdlib.h>

struct Process {

int id;

int burst\_time;

int waiting\_time;

int turnaround\_time;

};

int compare(const void \*a, const void \*b) {

return ((struct Process \*)a)->burst\_time - ((struct Process \*)b)->burst\_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 burst time for process %d: ", i + 1);

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

}

qsort(processes, n, sizeof(struct Process), compare);

processes[0].waiting\_time = 0;

processes[0].turnaround\_time = processes[0].burst\_time;

total\_turnaround\_time = processes[0].turnaround\_time;

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

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

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

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

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

}

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

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

printf("%d\t%d\t\t%d\t\t%d\n", processes[i].id, 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:

