PRIYANGA M **231901037**

EX.NO:6B
DATE:26.02.204

SHORTEST JOB FIRST

Aim: To implement the Shortest Job First (SJF) scheduling technique

Algorithm:
1. Declare the structure and its elements.
2. Get a number of processes as input from the user.
3. Read the process name, arrival time and burst time
4. Initialize waiting time, turnaround time & flag of read processes to zero.
5.Sort based on burst time of all processes in ascending order
6. Calculate the waiting time and turnaround time for each process.
7. Calculate the average waiting time and average turnaround time.
8. Display the results.

Program Code:

```
#include <stdio.h>
int main() {
        int n;
        // Step 1: Get the number of processes
        printf("Enter the number of processes: ");
        scanf("%d", &n);
        int burst_time[n], waiting_time[n], turnaround_time[n], process_order[n];
        // Step 2: Read the burst time for each process
        printf("Enter the burst time of the processes: ");
        for (int i = 0; i < n; i++) {
        scanf("%d", &burst_time[i]);
        process_order[i] = i + 1;  // Store the process number for display
        }
        // Step 3: Sort burst time in ascending order (SJF algorithm)
        for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
        if (burst_time[i] > burst_time[j]) {
                // Swap burst times
                int temp = burst_time[i];
                burst_time[i] = burst_time[j];
```

```
                burst_time[j] = temp;

                // Swap process order to maintain correct process sequence
                temp = process_order[i];
                process_order[i] = process_order[j];
                process_order[j] = temp;
            }
            }
            }
            // Initialize waiting time and turnaround time
            waiting_time[0] = 0;
            turnaround_time[0] = burst_time[0];
            // Step 4: Calculate waiting time and turnaround time for each process
            int total_waiting_time = 0;
            int total_turnaround_time = 0;
            // Calculate waiting time for each process
            for (int i = 1; i < n; i++) {
            waiting_time[i] = burst_time[i - 1] + waiting_time[i - 1];
            }
            // Calculate turnaround time for each process
            for (int i = 0; i < n; i++) {
            turnaround_time[i] = burst_time[i] + waiting_time[i];
            }
            // Step 5: Display the results
            printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
            for (int i = 0; i < n; i++) {
            printf("%d\t\t%d\t\t%d\t\t%d\n", process_order[i], burst_time[i], waiting_time[i],
turnaround_time[i]);
            total_waiting_time += waiting_time[i];
            total_turnaround_time += turnaround_time[i];
            }
            // Step 6: Calculate and display average waiting time and turnaround time
            float avg_waiting_time = (float)total_waiting_time / n;
            float avg_turnaround_time = (float)total_turnaround_time / n;
            printf("\nAverage Waiting Time: %.2f\n", avg_waiting_time);
            printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);
            return 0;
}
```

Output:

```
┌──(student㉿kali)-[~]
└─$ vi sjf.c

┌──(student㉿kali)-[~]
└─$ gcc sjf.c -o sjf

┌──(student㉿kali)-[~]
└─$ ./sjf
Enter the number of processes: 4
Enter the burst time of the processes: 8 4 9 5

Process Burst Time      Waiting Time    Turnaround Time
2               4               0               4
4               5               4               9
1               8               9               17
3               9               17              26

Average Waiting Time: 7.50
Average Turnaround Time: 14.00
```

Result: Hence, average waiting time and average turnaround time has been calculated using SJF algorithm.