NAME: G. GANESH

REG NO: 192373008

EXERICSE-7

Construct a C program to implement a non-preemptive SJF algorithm.

Aim:

To construct a C program to implement the non-preemptive Shortest Job First (SJF) scheduling algorithm, where processes are executed based on the shortest burst time.

Algorithm:

1. Input the number of processes and their burst times and arrival times.
2. Sort processes based on arrival time. For processes arriving at the same time, use burst time to determine the order.
3. Start executing processes:
    - At each step, select the process with the shortest burst time among the arrived processes.
    - Execute the process and update the current time.
4. Calculate waiting time and turnaround time for each process.
5. Compute the average waiting time and turnaround time.
6. Display the process execution order, waiting times, turnaround times, and averages.

Procedure:

1. Read the number of processes, burst times, and arrival times.
2. Sort the processes based on their arrival times.
3. Implement non-preemptive SJF by selecting the process with the shortest burst time among the ready processes.
4. Calculate waiting and turnaround times for each process.
5. Compute the average waiting and turnaround times.
6. Display the results.

Code:

```
#include <stdio.h>

int main() {

    int n, i, j;

    printf("Enter the number of processes: ");

    scanf("%d", &n);

    int burst_time[n], arrival_time[n], waiting_time[n], turnaround_time[n];

    int completed[n], current_time = 0, completed_count = 0;
```

```
    float avg_waiting_time = 0, avg_turnaround_time = 0;
    printf("Enter the arrival times and burst times of the processes:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d - Arrival Time: ", i + 1);
        scanf("%d", &arrival_time[i]);
        printf("Process %d - Burst Time: ", i + 1);
        scanf("%d", &burst_time[i]);
        completed[i] = 0; // Mark all processes as not completed initially
    }
    while (completed_count < n) {
        int shortest_index = -1, min_burst_time = 1e9;
        for (i = 0; i < n; i++) {
            if (!completed[i] && arrival_time[i] <= current_time && burst_time[i] < min_burst_time) {
                shortest_index = i;
                min_burst_time = burst_time[i];
            }
        }
        if (shortest_index == -1) {
            current_time++;
            continue;
        }
        current_time += burst_time[shortest_index];
        turnaround_time[shortest_index] = current_time - arrival_time[shortest_index];
        waiting_time[shortest_index] = turnaround_time[shortest_index] - burst_time[shortest_index];
        completed[shortest_index] = 1;
        completed_count++;
    }
    for (i = 0; i < n; i++) {
        avg_waiting_time += waiting_time[i];
```

```
        avg_turnaround_time += turnaround_time[i];
    }
    avg_waiting_time /= n;
    avg_turnaround_time /= n;
    printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, arrival_time[i], burst_time[i],
waiting_time[i], turnaround_time[i]);
    }
    printf("\nAverage Waiting Time: %.2f\n", avg_waiting_time);
    printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);
    return 0;
}
```

Result:

The preemptive priority scheduling program successfully executes the process with the highest priority at each time unit, calculates waiting and turnaround times, and computes averages. The program displays the process execution details and timing information.

Output:

```
Enter the number of processes: 4
Enter the arrival times and burst times of the processes:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 6
Process 2 - Arrival Time: 1
Process 2 - Burst Time: 8
Process 3 - Arrival Time: 2
Process 3 - Burst Time: 7
Process 4 - Arrival Time: 3
Process 4 - Burst Time: 3

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

Average Waiting Time: 6.25
Average Turnaround Time: 12.25
```