NAME: G. GANESH

REG NO: 192373008

EXERICSE-4

Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.

Aim:

To construct a CPU scheduling program in C using the Shortest Job Next (SJN) scheduling algorithm, where the process with the smallest execution time is executed next.

Algorithm:

1. Input the number of processes and their burst times.
2. Initialize all processes as unvisited (not yet executed).
3. Select the process with the smallest burst time from the set of unvisited processes to execute next.
4. Calculate the waiting time and turnaround time for the selected process.
5. Repeat until all processes are executed.
6. Compute the average waiting time and turnaround time.
7. Display the process execution order, waiting times, turnaround times, and averages.

Procedure:

1. Read the burst times for all processes.
2. Implement SJN by selecting the process with the smallest burst time iteratively.
3. Track waiting times and turnaround times during execution.
4. Calculate average waiting and turnaround times.
5. Display the results.

Code:

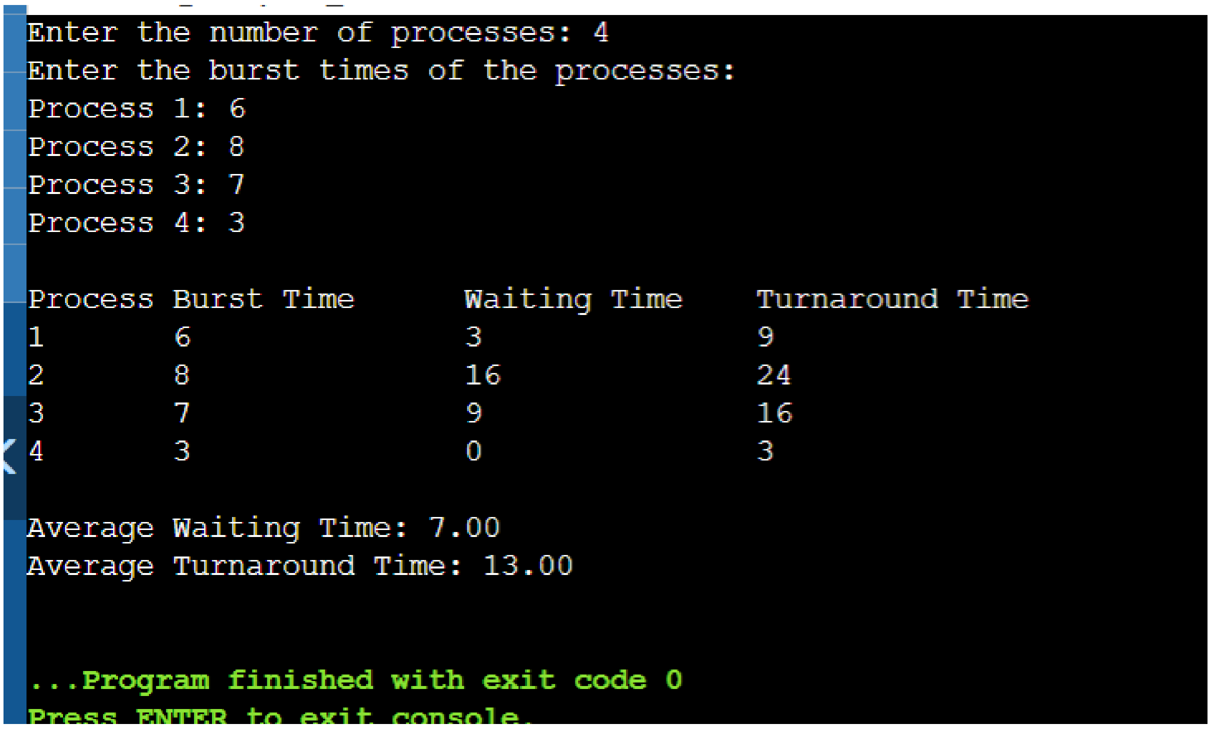
```
#include <stdio.h>

int main() {

    int n, i, j, completed = 0, time = 0, min_index;

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

    scanf("%d", &n);

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

    float avg_waiting_time = 0, avg_turnaround_time = 0;

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

    for (i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &burst_time[i]);
        visited[i] = 0; // Mark all processes as unvisited initially
    }
    while (completed < n) {
        int min_burst_time = 1e9;
        min_index = -1;
        for (i = 0; i < n; i++) {
            if (!visited[i] && burst_time[i] < min_burst_time) {
                min_burst_time = burst_time[i];
                min_index = i;
            }
        }
        if (min_index != -1) {
            waiting_time[min_index] = time;
            time += burst_time[min_index];
            turnaround_time[min_index] = time;
            visited[min_index] = 1;
            completed++;
        }
    }
    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\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\n", i + 1, 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 SJN scheduling program successfully selects the process with the smallest burst time for execution, calculates waiting and turnaround times, and computes the averages. The results include the order of execution and relevant timing details.

Output:



```
Enter the number of processes: 4
Enter the burst times of the processes:
Process 1: 6
Process 2: 8
Process 3: 7
Process 4: 3

Process Burst Time      Waiting Time    Turnaround Time
1       6               3               9
2       8               16              24
3       7               9               16
4       3               0               3

Average Waiting Time: 7.00
Average Turnaround Time: 13.00


...Program finished with exit code 0
Press ENTER to exit console.
```