Name: Kushal Kishor Shankhapal **Date:** 30/07/2023
Roll No: 56 **Subject:** OS Lab

SJF: Shortest Job First

Code:

```
#include <stdio.h>
#include <limits.h>  // For INT_MAX to represent a large number

#define MAX_PROCESSES 10  // Define the maximum number of processes
supported

int main() {
    // Arrays to store arrival times, burst times, and temporary burst
times
    int arrival_time[MAX_PROCESSES], burst_time[MAX_PROCESSES],
temp[MAX_PROCESSES];
    int remaining_time[MAX_PROCESSES];  // Array to track the remaining
burst times for each process
    int completion_time[MAX_PROCESSES], waiting_time[MAX_PROCESSES],
turnaround_time[MAX_PROCESSES];
    int i, smallest, count = 0, time, limit;
    int total_wait_time = 0, total_turnaround_time = 0;
    float average_waiting_time, average_turnaround_time;

    // Prompt the user to enter the number of processes
    printf("\nEnter the Total Number of Processes (max %d):\t",
MAX_PROCESSES);
    scanf("%d", &limit);

    // Check if the number of processes exceeds the maximum allowed
    if (limit > MAX_PROCESSES) {
        printf("Number of processes cannot exceed %d\n", MAX_PROCESSES);
        return 1;
    }

    // Read the arrival and burst times for each process
    printf("\nEnter Details of %d Processes\n", limit);
    for(i = 0; i < limit; i++) {
        printf("\nProcess %d:\n", i + 1);
        printf("Enter Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Enter Burst Time:\t");
        scanf("%d", &burst_time[i]);
        remaining_time[i] = burst_time[i];  // Initialize remaining time
with the burst time
        temp[i] = burst_time[i];  // Store the original burst time for
later calculations
    }

    int completed[MAX_PROCESSES] = {0};  // Array to keep track of which
processes have been completed

    // Main loop to simulate time and schedule processes
    for(time = 0; count != limit; time++) {
        smallest = -1;  // Initialize smallest as -1 to indicate no
process selected yet
```

```
        // Find the process with the smallest remaining burst time that
has arrived and is not yet completed
        for(i = 0; i < limit; i++) {
            if (arrival_time[i] <= time && !completed[i]) {
                if (smallest == -1 || remaining_time[i] <
remaining_time[smallest]) {
                    smallest = i;  // Update smallest to the current
process with the shortest remaining time
                }
            }
        }

        if (smallest != -1) {
            remaining_time[smallest]--;  // Execute the selected process
for one unit of time
            if (remaining_time[smallest] == 0) {
                // Process is completed
                count++;
                completed[smallest] = 1;  // Mark the process as completed
                completion_time[smallest] = time + 1;  // Calculate
completion time of the process
                waiting_time[smallest] = completion_time[smallest] -
arrival_time[smallest] - temp[smallest];
                turnaround_time[smallest] = completion_time[smallest] -
arrival_time[smallest];
                total_wait_time += waiting_time[smallest];  // Accumulate
total waiting time
                total_turnaround_time += turnaround_time[smallest];  //
Accumulate total turnaround time
            }
        }
    }

    // Calculate average waiting time and turnaround time
    average_waiting_time = (float)total_wait_time / limit;
    average_turnaround_time = (float)total_turnaround_time / limit;

    // Print the process details including arrival time, burst time,
completion time, waiting time, and turnaround time
    printf("\nProcess No\tAT\tBT\tCT\tTAT\tWT\n");
    for(i = 0; i < limit; i++) {
        printf("%d\t\t%d\t%d\t%d\t%d\t%d\n", i + 1, arrival_time[i],
temp[i], completion_time[i], turnaround_time[i], waiting_time[i]);
    }

    // Print the average waiting time and average turnaround time
    printf("\nAverage Waiting Time:\t%f", average_waiting_time);
    printf("\nAverage Turnaround Time:\t%f\n", average_turnaround_time);

    return 0;
}
```

Output:

```
pl-17@pl17-OptiPlex-3020:~$ cd Kushal_Files
pl-17@pl17-OptiPlex-3020:~/Kushal_Files$ g++ sjf_p1.cpp -o sjf.out
pl-17@pl17-OptiPlex-3020:~/Kushal_Files$ ./sjf.out


Enter the Total Number of Processes (max 10):       4

Enter Details of 4 Processes

Process 1:
Enter Arrival Time:    1
Enter Burst Time:      3

Process 2:
Enter Arrival Time:    2
Enter Burst Time:      4

Process 3:
Enter Arrival Time:    1
Enter Burst Time:      2

Process 4:
Enter Arrival Time:    4
Enter Burst Time:      4

Process No      AT      BT      CT      TAT     WT
1               1       3       6       5       2
2               2       4       10      8       4
3               1       2       3       2       0
4               4       4       14      10      6

Average Waiting Time:          3.000000
Average Turnaround Time:       6.250000
```