/*rr*/

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter Total Number of Processes: ");
    scanf("%d", &n);

    int wait_time = 0, ta_time = 0;
    int burst_time[n], remaining_burst_time[n];
    int time_slice;

    for (int i = 0; i < n; i++) {
        printf("Enter Burst Time for Process %d: ", i + 1);
        scanf("%d", &burst_time[i]);
        remaining_burst_time[i] = burst_time[i];
    }

    printf("Enter Time Slice (Quantum): ");
    scanf("%d", &time_slice);

    int total_time = 0;
    int completed_processes = 0;
    printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");

    int current_process = 0;
    while (completed_processes < n) {
        if (remaining_burst_time[current_process] > 0) {
            int execution_time;
            if (remaining_burst_time[current_process] > time_slice) {
                execution_time = time_slice;
```

```
        } else {
            execution_time = remaining_burst_time[current_process];
        }

        total_time += execution_time;
        remaining_burst_time[current_process] -= execution_time;

        if (remaining_burst_time[current_process] == 0) {
            completed_processes++;

            int turnaround_time = total_time;
            int waiting_time = turnaround_time - burst_time[current_process];

            printf("%d\t\t%d\t\t%d\t\t%d\n", current_process + 1, burst_time[current_process],
                   turnaround_time, waiting_time);

            wait_time += waiting_time;
            ta_time += turnaround_time;
        }

        // Move to the next process in a circular manner
        current_process = (current_process + 1) % n;
    } else {
        // If the process has already completed, move to the next process
        current_process = (current_process + 1) % n;
    }
}

float average_wait_time = (float)wait_time / n;
float average_turnaround_time = (float)ta_time / n;
```

```
    printf("\nAverage Waiting Time: %f", average_wait_time);
    printf("\nAverage Turnaround Time: %f\n", average_turnaround_time);

    return 0;
}
```