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

RR: Round Robin

Code:

```
#include <stdio.h>

#define MAX_PROCESSES 10  // Define a constant for the maximum number of
processes

int main() {
    int i, limit, total_time = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0;
    int arrival_time[MAX_PROCESSES], burst_time[MAX_PROCESSES],
temp_burst_time[MAX_PROCESSES];
    int remaining_processes = 0, counter = 0;
    float average_wait_time, average_turnaround_time;

    // Input number of processes
    printf("Enter Total Number of Processes (max %d):\n\t",
MAX_PROCESSES);
    scanf("%d", &limit);

    // Input arrival and burst times for each process
    for (i = 0; i < limit; i++) {
        printf("Enter Details of Process[%d]\n", i + 1);

        printf("Arrival Time:\t");
        scanf("%d", &arrival_time[i]);

        printf("Burst Time:\t");
        scanf("%d", &burst_time[i]);

        temp_burst_time[i] = burst_time[i];  // Copy burst times for
processing
    }

    // Input time quantum
    printf("Enter Time Quantum:\n\t");
    scanf("%d", &time_quantum);

    printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");

    // Main Round Robin scheduling loop
    for (i = 0, remaining_processes = limit; remaining_processes != 0;) {
        if (temp_burst_time[i] <= time_quantum && temp_burst_time[i] > 0)
{
            total_time += temp_burst_time[i];
            temp_burst_time[i] = 0;
            counter = 1;
        } else if (temp_burst_time[i] > 0) {
            temp_burst_time[i] -= time_quantum;
            total_time += time_quantum;
        }

        // Check if the process is complete
        if (temp_burst_time[i] == 0 && counter == 1) {
```

```
            remaining_processes--;

            // Calculate turnaround time and waiting time
            int turnaround = total_time - arrival_time[i];
            int wait = turnaround - burst_time[i];
            wait_time += wait;
            turnaround_time += turnaround;

            // Print process details
            printf("Process[%d]\t\t%d\t%d\t\t%d\n", i + 1, burst_time[i],
turnaround, wait);

            counter = 0;
        }

        // Move to the next process
        if (i == limit - 1) {
            i = 0;
        } else if (arrival_time[i + 1] <= total_time) {
            i++;
        } else {
            i = 0;
        }
    }

    // Calculate average waiting time and turnaround time
    average_wait_time = (float)wait_time / limit;
    average_turnaround_time = (float)turnaround_time / limit;

    // Print average times
    printf("\nAverage Waiting Time:\t%f", average_wait_time);
    printf("\nAverage Turnaround Time:\t%f", average_turnaround_time);

    return 0;
}
```

Output:

```
pl-17@pl17-OptiPlex-3020:~/Kushal_Files$ gcc RR_1.c
pl-17@pl17-OptiPlex-3020:~/Kushal_Files$ ./a.out
Enter Total Number of Processes (max 10):
        4
Enter Details of Process[1]
Arrival Time:  0
Burst Time:    5
Enter Details of Process[2]
Arrival Time:  1
Burst Time:    4
Enter Details of Process[3]
Arrival Time:  2
Burst Time:    2
Enter Details of Process[4]
Arrival Time:  4
Burst Time:    1
Enter Time Quantum:
```

2

Process ID	Burst Time	Turnaround Time	Waiting Time
Process[3]	2	4	2
Process[4]	1	3	2
Process[2]	4	10	6
Process[1]	5	12	7

Average Waiting Time: 4.250000
Average Turnaround Time: 7.250000