**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