**NAME: G. GANESH**

**REG NO: 192373008**

# EXERICSE-8

**Construct a C program to simulate Round Robin scheduling algorithm with C.**

**Aim:**

To construct a C program to simulate the Round Robin (RR) scheduling algorithm, where processes are executed in a cyclic manner using a fixed time quantum.

**Algorithm:**

1. Input the number of processes, their burst times, and the time quantum.
2. Initialize the ready queue and keep track of remaining burst times for all processes.
3. While there are incomplete processes:
    - Execute the process at the front of the ready queue for a duration equal to the time quantum or its remaining burst time.
    - If the process completes, record its completion time.
    - If the process is not complete, move it to the back of the queue.
4. Calculate waiting and turnaround times for all processes.
5. Compute the average waiting time and turnaround time.
6. Display the scheduling order, waiting times, turnaround times, and averages.

**Procedure:**

1. Read the number of processes, their burst times, and the time quantum.
2. Implement the Round Robin scheduling algorithm using a circular queue.
3. Calculate the waiting and turnaround times for each process.
4. Compute and display the average waiting time and turnaround time.

**Code:**

```
#include <stdio.h>

int main() {
    int n, i, time = 0, completed = 0, quantum;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int burst_time[n], remaining_time[n], arrival_time[n];
    int waiting_time[n], turnaround_time[n], is_completed[n];
```

```
float avg_waiting_time = 0, avg_turnaround_time = 0;
printf("Enter the arrival times and burst times of the processes:\n");
for (i = 0; i < n; i++) {
    printf("Process %d - Arrival Time: ", i + 1);
    scanf("%d", &arrival_time[i]);
    printf("Process %d - Burst Time: ", i + 1);
    scanf("%d", &burst_time[i]);
    remaining_time[i] = burst_time[i];
    waiting_time[i] = 0;
    is_completed[i] = 0;
}
printf("Enter the time quantum: ");
scanf("%d", &quantum);
while (completed < n) {
    int done = 1;
    for (i = 0; i < n; i++) {
        if (remaining_time[i] > 0) {
            done = 0;
            if (remaining_time[i] > quantum) {
                time += quantum;
                remaining_time[i] -= quantum;
            } else {
                time += remaining_time[i];
                waiting_time[i] = time - arrival_time[i] - burst_time[i];
                turnaround_time[i] = time - arrival_time[i];
                remaining_time[i] = 0;
                completed++;
            }
        }
    }
```

```
        if (done) break;
    }
    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\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, arrival_time[i], 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 Round Robin scheduling program successfully schedules processes in a cyclic manner with a fixed time quantum, calculates waiting and turnaround times, and computes averages. The program displays the process details and their respective timing information.

**Output:**

```
Enter the number of processes: 3
Enter the arrival times and burst times of the processes:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 5
Process 2 - Arrival Time: 1
Process 2 - Burst Time: 4
Process 3 - Arrival Time: 2
Process 3 - Burst Time: 3
Enter the time quantum: 2

Process Arrival Time    Burst Time      Waiting Time    Turnaround Time
1       0               5               7               12
2       1               4               5               9
3       2               3               6               9

Average Waiting Time: 6.00
Average Turnaround Time: 10.00


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