**EXPERIMENT - 8**

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

#include <stdio.h>

#define MAX 100

struct Process {

int id;

int arrival\_time;

int burst\_time;

int remaining\_time;

int completion\_time;

int waiting\_time;

int turnaround\_time;

};

int main() {

struct Process p[MAX];

int n, time = 0, completed = 0, tq;

float total\_wt = 0, total\_tat = 0;

int queue[MAX], front = 0, rear = 0;

int visited[MAX] = {0};

printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter arrival time and burst time for each process:\n");

for (int i = 0; i < n; i++) {

p[i].id = i + 1;

printf("Process %d: ", p[i].id);

scanf("%d%d", &p[i].arrival\_time, &p[i].burst\_time);

p[i].remaining\_time = p[i].burst\_time;

}

printf("Enter Time Quantum: ");

scanf("%d", &tq);

queue[rear++] = 0;

visited[0] = 1;

printf("\n--- Gantt Chart ---\n");

while (completed < n) {

int idx = queue[front++];

if (p[idx].remaining\_time > 0) {

printf("| P%d ", p[idx].id);

if (p[idx].remaining\_time <= tq) {

time += p[idx].remaining\_time;

p[idx].remaining\_time = 0;

p[idx].completion\_time = time;

p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;

p[idx].waiting\_time = p[idx].turnaround\_time - p[idx].burst\_time;

total\_wt += p[idx].waiting\_time;

total\_tat += p[idx].turnaround\_time;

completed++;

} else {

time += tq;

p[idx].remaining\_time -= tq;

}

for (int i = 0; i < n; i++) {

if (!visited[i] && p[i].arrival\_time <= time) {

queue[rear++] = i;

visited[i] = 1;

}

}

if (p[idx].remaining\_time > 0) {

queue[rear++] = idx;

}

}

if (front == rear) {

for (int i = 0; i < n; i++) {

if (p[i].remaining\_time > 0) {

time = p[i].arrival\_time;

queue[rear++] = i;

visited[i] = 1;

break;

}

}

}

}

printf("|\n");

printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\n");

for (int i = 0; i < n; i++) {

printf("P%d\t%d\t%d\t%d\t%d\t%d\n",

p[i].id,

p[i].arrival\_time,

p[i].burst\_time,

p[i].completion\_time,

p[i].turnaround\_time,

p[i].waiting\_time);

}

printf("\nAverage Waiting Time: %.2f\n", total\_wt / n);

printf("Average Turnaround Time: %.2f\n", total\_tat / n);

return 0;

}

SAMPLE INPUT:

Enter the number of processes: 4

Process 1: 0 5

Process 2: 1 4

Process 3: 2 2

Process 4: 3 1

Enter Time Quantum: 2

SAMPLE OUTPUT:

| P1 | P2 | P3 | P4 | P1 | P2 | P1 |

Process AT BT CT TAT WT

P1 0 5 11 11 6

P2 1 4 10 9 5

P3 2 2 6 4 2

P4 3 1 5 2 1

Average Waiting Time: 3.50

Average Turnaround Time: 6.50