**Experiment No 6**

**Aim -: To write a program to implement CPU scheduling for Round Robin scheduling**

#include<stdio.h>

Struct process {

Int pid;

Int arrival\_time;

Int burst\_time;

Int remaining\_time;

Int waiting\_time;

Int turnaround\_time;

};

Void main() {

Int n, I, quantum, remaining\_processes;

Struct process proc[10];

Float total\_waiting\_time = 0, total\_turnaround\_time = 0;

Printf(“Enter the number of processes: “);

Scanf(“%d”, &n);

// Input process details

For(i=0; i<n; i++) {

Printf(“\nEnter the process id, arrival time, and burst time for process %d: “, i+1);

Scanf(“%d%d%d”, &proc[i].pid, &proc[i].arrival\_time, &proc[i].burst\_time);

Proc[i].remaining\_time = proc[i].burst\_time;

}

Printf(“\nEnter the quantum: “);

Scanf(“%d”, &quantum);

// Simulate Round Robin scheduling

Remaining\_processes = n;

Int time = 0;

While(remaining\_processes > 0) {

For(i=0; i<n; i++) {

If(proc[i].remaining\_time > 0) {

If(proc[i].remaining\_time <= quantum) {

Time += proc[i].remaining\_time;

Proc[i].turnaround\_time = time – proc[i].arrival\_time;

Proc[i].waiting\_time = proc[i].turnaround\_time – proc[i].burst\_time;

Total\_waiting\_time += proc[i].waiting\_time;

Total\_turnaround\_time += proc[i].turnaround\_time;

Proc[i].remaining\_time = 0;

Remaining\_processes--;

}

Else {

Time += quantum;

Proc[i].remaining\_time -= quantum;

}

}

}

}

// Print process details and scheduling metrics

Printf(“\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time”);

For(i=0; i<n; i++) {

Printf(“\n%d\t\t%d\t\t%d\t\t%d\t\t%d”, proc[i].pid, proc[i].arrival\_time, proc[i].burst\_time, proc[i].waiting\_time, proc[i].turnaround\_time);

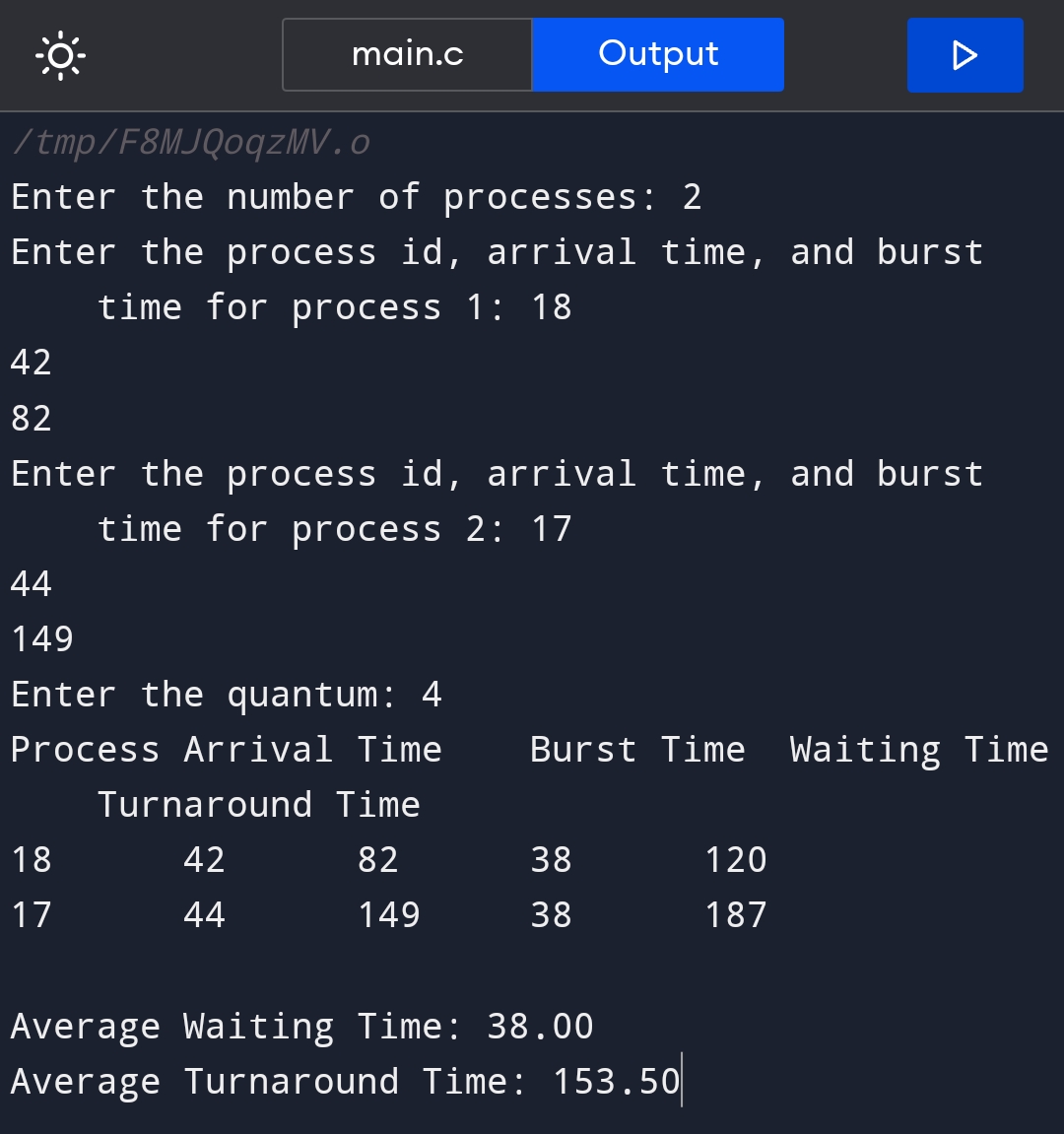
}

Printf(“\n\nAverage Waiting Time: %.2f”, total\_waiting\_time/n);

Printf(“\nAverage Turnaround Time: %.2f”, total\_turnaround\_time/n);

}

**Output -:**

****