#include <stdio.h>

struct Process {

int pid; // Process ID

int burst; // Process burst time

int remaining; // Remaining burst time

};

void roundRobinScheduling(struct Process processes[], int n, int timeQuantum) {

int count = n; // The number of processes

int currentTime = 0; // Current time

int i, idx;

float avgWaitingTime = 0, avgTurnaroundTime = 0;

printf("\nGantt Chart:\n");

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

printf("| Process ID | Burst Time | Turnaround Time | Waiting Time |\n");

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

// Execute until all processes are done

while (count != 0) {

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

if (processes[i].remaining > 0) {

// Process executes for the given time quantum

if (processes[i].remaining > timeQuantum) {

currentTime += timeQuantum;

processes[i].remaining -= timeQuantum;

} else {

currentTime += processes[i].remaining;

processes[i].remaining = 0;

}

// Display the execution of the process in Gantt chart

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

processes[i].pid, currentTime, currentTime, currentTime - processes[i].burst);

// Calculate average waiting and turnaround time

avgWaitingTime += currentTime - processes[i].burst;

avgTurnaroundTime += currentTime;

if (processes[i].remaining == 0) {

count--; // Decrease the number of remaining processes

}

}

}

}

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

// Display average waiting and turnaround time

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

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

}

int main() {

int n, i, timeQuantum;

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

scanf("%d", &n);

struct Process processes[n];

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

printf("Enter the burst time for process P%d: ", i+1);

scanf("%d", &processes[i].burst);

processes[i].remaining = processes[i].burst;

processes[i].pid = i+1;

}

printf("Enter the time quantum: ");

scanf("%d", &timeQuantum);

roundRobinScheduling(processes, n, timeQuantum);

return 0;

}