// Program to compute finish time, turn around time & waiting time for round robin

#include <stdio.h>

struct Process {

int pid; // Process ID

int arrivalTime; // Arrival time

int burstTime; // Burst time

int remainingTime; // Remaining burst time

int finishTime; // Finish time

int turnAroundTime; // Turnaround time

int waitingTime; // Waiting time

};

void roundRobin(struct Process processes[], int n, int quantum) {

int currentTime = 0;

int completed = 0;

int timeQuantum = quantum;

while (completed < n) {

int done = 1;

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

// Check if process has remaining time and has arrived

if (processes[i].remainingTime > 0 && processes[i].arrivalTime <= currentTime) {

done = 0;

// If remaining time is less than or equal to time quantum, process will finish

if (processes[i].remainingTime <= timeQuantum) {

currentTime += processes[i].remainingTime;

processes[i].finishTime = currentTime;

processes[i].turnAroundTime = processes[i].finishTime - processes[i].arrivalTime;

processes[i].waitingTime = processes[i].turnAroundTime - processes[i].burstTime;

processes[i].remainingTime = 0;

completed++;

} else {

// Process runs for the time quantum

processes[i].remainingTime -= timeQuantum;

currentTime += timeQuantum;

}

}

}

// If all processes are done

if (done) {

currentTime++;

}

}

}

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

printf("PID\tArrival\tBurst\tFinish\tTurnaround\tWaiting\n");

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

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

processes[i].pid,

processes[i].arrivalTime,

processes[i].burstTime,

processes[i].finishTime,

processes[i].turnAroundTime,

processes[i].waitingTime);

}

float totalTurnAroundTime = 0, totalWaitingTime = 0;

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

totalTurnAroundTime += processes[i].turnAroundTime;

totalWaitingTime += processes[i].waitingTime;

}

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

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

}

int main() {

int n, quantum;

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

scanf("%d", &n);

struct Process processes[n];

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

processes[i].pid = i + 1;

printf("Enter arrival time and burst time for process %d: ", processes[i].pid);

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

processes[i].remainingTime = processes[i].burstTime;

}

printf("Enter the time quantum: ");

scanf("%d", &quantum);

roundRobin(processes, n, quantum);

displayResults(processes, n);

return 0;

}