// Program to compute finish time, time around time & waiting time for first come first serve.

#include <stdio.h>

struct Process {

int pid; // Process ID

int arrivalTime;

int burstTime;

int finishTime;

int turnAroundTime;

int waitingTime;

};

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

int currentTime = 0;

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

// If the process arrives after the current time, update the current time to the arrival time of the process

if (currentTime < processes[i].arrivalTime) {

currentTime = processes[i].arrivalTime;

}

// Finish time is the current time plus the burst time

processes[i].finishTime = currentTime + processes[i].burstTime;

// Turnaround time is finish time minus arrival time

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

// Waiting time is turnaround time minus burst time

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

// Update current time to finish time of the current process

currentTime = processes[i].finishTime;

}

}

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;

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);

}

// Sort processes by arrival time (FCFS scheduling)

for (int i = 0; i < n - 1; i++) {

for (int j = i + 1; j < n; j++) {

if (processes[i].arrivalTime > processes[j].arrivalTime) {

struct Process temp = processes[i];

processes[i] = processes[j];

processes[j] = temp;

}

}

}

calculateTimes(processes, n);

displayResults(processes, n);

return 0;

}