#include <stdio.h>

struct Process {

int pid;

int burstTime;

int arrivalTime;

int priority;

int waitingTime;

int turnaroundTime;

int executed;

};

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

int totalWaitTime = 0, totalTurnaroundTime = 0;

int i;

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

processes[i].turnaroundTime = processes[i].burstTime + processes[i].waitingTime;

totalWaitTime += processes[i].waitingTime;

totalTurnaroundTime += processes[i].turnaroundTime;

}

printf("Process\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");

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

printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burstTime, processes[i].priority, processes[i].waitingTime, processes[i].turnaroundTime);

}

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

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

}

int findHighestPriority(struct Process processes[], int n, int time) {

int highest = -1;

int maxPriority = -1;

int i;

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

if (processes[i].arrivalTime <= time && processes[i].executed == 0 && processes[i].priority > maxPriority) {

maxPriority = processes[i].priority;

highest = i;

}

}

return highest;

}

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

int currentTime = 0;

int completed = 0;

while (completed != n) {

int highest = findHighestPriority(processes, n, currentTime);

if (highest == -1) {

currentTime++;

continue;

}

processes[highest].waitingTime = currentTime - processes[highest].arrivalTime;

currentTime += processes[highest].burstTime;

processes[highest].executed = 1;

completed++;

}

calculateTimes(processes, n);

}

int main() {

int n,i;

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

scanf("%d", &n);

struct Process processes[n];

printf("Enter burst times and priorities for each process:\n");

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

processes[i].pid = i + 1;

printf("Burst time P %d: ", i + 1);

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

printf("Priority P %d: ", i + 1);

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

processes[i].arrivalTime = 0;

processes[i].executed = 0;

}

schedulePriority(processes, n);

    return 0;

}