#include <stdio.h>

struct Process {

int processID;

int arrivalTime;

int burstTime;

int remainingTime;

};

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

int timeQuantum = 2;

int currentTime = 0;

int waitingTime[n];

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

waitingTime[i] = 0;

}

while (1) {

int allProcessesComplete = 1;

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

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

allProcessesComplete = 0

int executeTime = (processes[i].remainingTime < timeQuantum) ? processes[i].remainingTime : timeQuantum;

processes[i].remainingTime -= executeTime;

waitingTime[i] += currentTime - processes[i].arrivalTime;

currentTime += executeTime;

}

}

if (allProcessesComplete)

break;

}

float averageWaitingTime = 0;

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

averageWaitingTime += waitingTime[i];

}

averageWaitingTime /= n;

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

}

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].processID = i + 1;

printf("Enter arrival time for Process %d: ", i + 1);

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

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

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

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

}

roundRobinScheduling(processes, n);

return 0;

}