import java.util.\*;

// Define a class named FCFS

public class FCFS {

// Define the main method

public static void main(String args[]) {

// Create a Scanner object to read input from the user

Scanner sc = new Scanner(System.in);

// Ask the user to enter the number of processes

System.out.println("enter no of process: ");

int n = sc.nextInt(); // Read the input number of processes

// Define arrays to store process details for each process

int pid[] = new int[n]; // process ids

int ar[] = new int[n]; // arrival times

int bt[] = new int[n]; // burst or execution times

int ct[] = new int[n]; // completion times

int ta[] = new int[n]; // turn around times

int wt[] = new int[n]; // waiting times

int temp; // A temporary variable used for swapping

// Initialize average waiting time and average turnaround time to 0

float avgwt=0,avgta=0;

// Input process details from the user

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

// Ask for the arrival time of the current process

System.out.println("enter process " + (i+1) + " arrival time: ");

ar[i] = sc.nextInt(); // Read the arrival time for the current process

// Ask for the burst time of the current process

System.out.println("enter process " + (i+1) + " burst time: ");

bt[i] = sc.nextInt(); // Read the burst time for the current process

pid[i] = i+1; // Assigning process IDs from 1 to n

}

// Sorting the processes based on arrival time using Bubble Sort

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

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

if( ar[j] > ar[j+1] ) {

// Swap the arrival time, burst time, and process IDs of the two processes

temp = ar[j];

ar[j] = ar[j+1];

ar[j+1] = temp;

temp = bt[j];

bt[j] = bt[j+1];

bt[j+1] = temp;

temp = pid[j];

pid[j] = pid[j+1];

pid[j+1] = temp;

}

}

}

// Finding the completion times of each process

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

if (i == 0) {

ct[i] = ar[i] + bt[i]; // If it's the first process, completion time is arrival time + burst time

} else {

if (ar[i] > ct[i-1]) {

ct[i] = ar[i] + bt[i]; // If the arrival time of the current process is after the completion of the previous process

} else {

ct[i] = ct[i-1] + bt[i]; // If the arrival time is before or equal to the completion time of the previous process

}

}

ta[i] = ct[i] - ar[i]; // Turnaround time = Completion time - Arrival time

wt[i] = ta[i] - bt[i]; // Waiting time = Turnaround time - Burst time

avgwt += wt[i]; // Summing up waiting times for calculating the average later

avgta += ta[i]; // Summing up turnaround times for calculating the average later

}

// Displaying the process details in tabular form

System.out.println("\npid arrival burst complete turn waiting");

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

System.out.println(pid[i] + " \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i]);

}

sc.close(); // Close the Scanner object as we no longer need user input

// Printing the average waiting time and average turnaround time

System.out.println("\naverage waiting time: "+ (avgwt/n));

System.out.println("average turnaround time:"+(avgta/n));

}

}