import java.util.Scanner;

class FCFS {

    public static void main(String args[]) {

        // Declaration of variables and arrays to store burst time, arrival time, etc.

        int burst\_time[], arrival\_time[], process[], waiting\_time[], tat[], i, j, n, totalWait = 0, totalTat = 0, totalTime = 0;

        float wait\_avg, tat\_avg, throughput;

        Scanner s = new Scanner(System.in); // Scanner for input

        // Taking the number of processes as input

        System.out.print("Enter number of processes: ");

        n = s.nextInt();

        // Initializing arrays based on number of processes

        process = new int[n];         // Process IDs

        burst\_time = new int[n];      // Burst time for each process

        arrival\_time = new int[n];    // Arrival time for each process

        waiting\_time = new int[n];    // Waiting time for each process

        tat = new int[n];             // Turnaround time for each process

        // Taking burst time and arrival time for each process

        System.out.println("\nEnter Arrival Time and Burst Time:");

        for (i = 0; i < n; i++) {

            System.out.print("Process[" + (i + 1) + "] Arrival Time: ");

            arrival\_time[i] = s.nextInt();

            System.out.print("Process[" + (i + 1) + "] Burst Time: ");

            burst\_time[i] = s.nextInt();

            process[i] = i + 1; // Assigning process numbers as 1, 2, 3, ...

        }

        // Sorting processes by arrival time to follow FCFS scheduling order

        for (i = 0; i < n; i++) {

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

                if (arrival\_time[i] > arrival\_time[j]) {

                    // Swap arrival times

                    int temp = arrival\_time[i];

                    arrival\_time[i] = arrival\_time[j];

                    arrival\_time[j] = temp;

                    // Swap burst times to maintain order with arrival time

                    temp = burst\_time[i];

                    burst\_time[i] = burst\_time[j];

                    burst\_time[j] = temp;

                    // Swap process numbers to keep process ID in correct order

                    temp = process[i];

                    process[i] = process[j];

                    process[j] = temp;

                }

            }

        }

        // Calculating waiting time for the first process (0 if it arrives at time 0)

        waiting\_time[0] = 0;

        totalTime = arrival\_time[0] + burst\_time[0]; // Initial completion time after first process

        tat[0] = burst\_time[0];  // Turnaround time for first process is its burst time

        // Calculating waiting time and turnaround time for each subsequent process

        for (i = 1; i < n; i++) {

            // If the total time so far is less than the next arrival, adjust to next arrival

            if (totalTime < arrival\_time[i]) {

                totalTime = arrival\_time[i];

            }

            // Waiting time = total time elapsed so far - arrival time of the process

            waiting\_time[i] = totalTime - arrival\_time[i];

            tat[i] = waiting\_time[i] + burst\_time[i];  // Turnaround time = waiting time + burst time

            // Adding to total waiting time and total turnaround time

            totalWait += waiting\_time[i];

            totalTat += tat[i];

            // Updating total time to account for this process's burst time

            totalTime += burst\_time[i];

        }

        // Calculate average waiting time and average turnaround time

        wait\_avg = (float) totalWait / n;

        tat\_avg = (float) totalTat / n;

        // Calculate throughput: number of processes / total time taken to complete all processes

        throughput = (float) n / totalTime;

        // Displaying the results

        System.out.println("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time");

        for (i = 0; i < n; i++) {

            System.out.println("P" + process[i] + "\t\t" + arrival\_time[i] + "\t\t" + burst\_time[i] +

                               "\t\t" + waiting\_time[i] + "\t\t" + tat[i]);

        }

        // Displaying the calculated averages and throughput

        System.out.printf("\nAverage Waiting Time: %.2f", wait\_avg);

        System.out.printf("\nAverage Turnaround Time: %.2f", tat\_avg);

        System.out.printf("\nThroughput: %.2f processes/unit time\n", throughput);

    }

}

// Enter number of processes: 3

// Enter Arrival Time and Burst Time:

// Process[1] Arrival Time: 0

// Process[1] Burst Time: 5

// Process[2] Arrival Time: 2

// Process[2] Burst Time: 3

// Process[3] Arrival Time: 4

// Process[3] Burst Time: 1

// Process  Arrival Time  Burst Time  Waiting Time  Turnaround Time

// P1   0   5   0   5

// P2   2   3   3   6

// P3   4   1   4   5

// Average Waiting Time: 2.33

// Average Turnaround Time: 5.33

// Throughput: 0.43 processes/unit time