import java.util.Scanner;

class SJF {

    public static void main(String args[]) {

        // Declaration of variables and arrays

        int burst\_time[], process[], waiting\_time[], tat[], i, j, n, total = 0, pos, temp;

        float wait\_avg, TAT\_avg;

        // Scanner object to take input from the user

        Scanner s = new Scanner(System.in);

        // Taking the number of processes as input

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

        n = s.nextInt();

        // Initializing arrays based on number of processes

        process = new int[n];         // Process IDs (P1, P2, P3, ...)

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

        System.out.println("\nEnter Burst time:");

        for (i = 0; i < n; i++) {

            System.out.print("\nProcess[" + (i + 1) + "]: ");

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

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

        }

        // Sorting processes based on burst time in ascending order (SJF)

        for (i = 0; i < n; i++) {

            pos = i;

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

                // Find the position of the shortest burst time process

                if (burst\_time[j] < burst\_time[pos])

                    pos = j;

            }

            // Swap burst time and corresponding process numbers to maintain order

            temp = burst\_time[i];

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

            burst\_time[pos] = temp;

            temp = process[i];

            process[i] = process[pos];

            process[pos] = temp;

        }

        // First process has 0 waiting time (it starts executing immediately)

        waiting\_time[0] = 0;

        // Calculate waiting time for the rest of the processes

        for (i = 1; i < n; i++) {

            waiting\_time[i] = 0;

            // Sum up burst times of the previous processes to calculate waiting time

            for (j = 0; j < i; j++) {

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

            }

            total += waiting\_time[i];  // Sum up all waiting times

        }

        // Calculating Average Waiting Time

        wait\_avg = (float) total / n;

        total = 0;  // Reset total for next calculations

        // Displaying Process details: Process, Burst time, Waiting time, Turnaround time

        System.out.println("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

        for (i = 0; i < n; i++) {

            // Turnaround Time = Burst Time + Waiting Time

            tat[i] = burst\_time[i] + waiting\_time[i];

            total += tat[i];  // Sum up all turnaround times

            // Print each process details

            System.out.println("\nP" + process[i] + "\t\t" + burst\_time[i] + "\t\t" + waiting\_time[i] + "\t\t" + tat[i]);

        }

        // Calculating Average Turnaround Time

        TAT\_avg = (float) total / n;

        // Displaying the average waiting time and turnaround time

        System.out.println("\n\nAverage Waiting Time: " + wait\_avg);

        System.out.println("Average Turnaround Time: " + TAT\_avg);

    }

}

// output

// Enter number of process: 4

// Enter Burst time:

// Process[1]: 6

// Process[2]: 8

// Process[3]: 7

// Process[4]: 3

// Process   Burst Time   Waiting Time  Turnaround Time

// P4   3   0   3

// P1   6   3   9

// P3   7   9   16

// P2   8   16    24

// Average Waiting Time: 7.00

// Average Turnaround Time: 13.00

//FORMULAS

//WAITING TIME=TURN AROUND TIME-BURST TIME

//TURN AROUND TIME=COMPLETION TIME-ARRIVAL TIME