import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.PriorityQueue;

class Process {

    int id;

    int arrivalTime;

    int burstTime;

    int remainingTime;

    int completionTime;

    int turnAroundTime;

    int waitingTime;

    Process(int id, int arrivalTime, int burstTime) {

        this.id = id;

        this.arrivalTime = arrivalTime;

        this.burstTime = burstTime;

        this.remainingTime = burstTime;

    }

}

public class SJFPreemptive {

    static void findAverageTime(List<Process> processes) {

        int currentTime = 0;

        int completed = 0;

        int n = processes.size();

        int totalTurnAroundTime = 0;

        int totalWaitingTime = 0;

        PriorityQueue<Process> minHeap = new PriorityQueue<>(Comparator.comparingInt(p -> p.remainingTime));

        processes.sort(Comparator.comparingInt(p -> p.arrivalTime));

        int i = 0;

        while (completed != n) {

            while (i < n && processes.get(i).arrivalTime <= currentTime) {

                minHeap.add(processes.get(i));

                i++;

            }

            if (minHeap.isEmpty()) {

                currentTime = processes.get(i).arrivalTime;

                continue;

            }

            Process currentProcess = minHeap.poll();

            currentProcess.remainingTime--;

            currentTime++;

            if (currentProcess.remainingTime == 0) {

                completed++;

                currentProcess.completionTime = currentTime;

                currentProcess.turnAroundTime = currentProcess.completionTime - currentProcess.arrivalTime;

                currentProcess.waitingTime = currentProcess.turnAroundTime - currentProcess.burstTime;

                totalTurnAroundTime += currentProcess.turnAroundTime;

                totalWaitingTime += currentProcess.waitingTime;

            } else {

                minHeap.add(currentProcess);

            }

        }

        System.out.println("Processes  Arrival Time  Burst Time  Completion Time  Turnaround Time  Waiting Time");

        for (Process p : processes) {

            System.out.printf("   %d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p.id, p.arrivalTime, p.burstTime, p.completionTime, p.turnAroundTime, p.waitingTime);

        }

        System.out.printf("Average turnaround time = %.2f\n", (float) totalTurnAroundTime / n);

        System.out.printf("Average waiting time = %.2f\n", (float) totalWaitingTime / n);

    }

    public static void main(String[] args) {

        List<Process> processes = new ArrayList<>();

        processes.add(new Process(1, 0, 6));

        processes.add(new Process(2, 1, 3));

        processes.add(new Process(3, 2, 7));

        processes.add(new Process(4, 3, 8));

        findAverageTime(processes);

    }

}