SRTF

import java.util.\*;

public class SRTF {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

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

int at[] = new int[n]; // Arrival times

int bt[] = new int[n]; // Burst times

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

int ta[] = new int[n]; // Turnaround times

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

int f[] = new int[n]; // Flag to check if process is completed

int k[] = new int[n]; // To store the original burst times for calculations

int i, st = 0, tot = 0; // st: system time, tot: total completed processes

float avgwt = 0, avgta = 0;

// Input for arrival and burst times

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

pid[i] = i + 1;

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

at[i] = sc.nextInt();

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

bt[i] = sc.nextInt();

k[i] = bt[i]; // Store the original burst time

f[i] = 0; // Mark the process as not completed

}

// SRTF scheduling starts

while (true) {

int min = 99, c = n; // c: index of the process to execute, min: minimum burst time

if (tot == n) // If all processes are completed, break the loop

break;

// Find the process with the shortest remaining burst time

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

if ((at[i] <= st) && (f[i] == 0) && (bt[i] < min)) {

min = bt[i];

c = i;

}

}

if (c == n) // If no process is ready to execute, increment system time

st++;

else {

bt[c]--; // Decrement the burst time of the selected process

st++; // Increment system time

if (bt[c] == 0) { // If process is completed

ct[c] = st; // Set completion time

f[c] = 1; // Mark process as completed

tot++; // Increment total completed processes

}

}

}

// Calculate turnaround time and waiting time

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

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

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

avgwt += wt[i]; // Accumulate total waiting time

avgta += ta[i]; // Accumulate total turnaround time

}

// Display process details

System.out.println("PID\tArrival\tBurst\tComplete\tTurnaround\tWaiting");

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

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

}

// Display average turnaround and waiting times

System.out.println("\nAverage Turnaround Time: " + (avgta / n));

System.out.println("Average Waiting Time: " + (avgwt / n));

sc.close();

}

}