#include <stdio.h>

struct Process {

int id;

int arrival\_time;

int burst\_time;

int completion\_time;

int turnaround\_time;

int waiting\_time;

};

void swap(struct Process \*a, struct Process \*b) {

struct Process temp = \*a;

\*a = \*b;

\*b = temp;

}

void sortProcessesByArrivalTime(struct Process processes[], int n) {

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

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

if (processes[j].arrival\_time > processes[j + 1].arrival\_time) {

swap(&processes[j], &processes[j + 1]);

}

}

}

}

void SJF(struct Process processes[], int n) {

sortProcessesByArrivalTime(processes, n);

int currentTime = 0;

printf("Gantt Chart:\n");

while (1) {

int shortestJobIndex = -1;

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

if (processes[i].arrival\_time <= currentTime && processes[i].burst\_time > 0) {

if (shortestJobIndex == -1 || processes[i].burst\_time < processes[shortestJobIndex].burst\_time) {

shortestJobIndex = i;

}

}

}

if (shortestJobIndex == -1) {

printf(" %d ", currentTime);

currentTime++;

continue;

}

printf(" %d | P%d ", currentTime, processes[shortestJobIndex].id);

processes[shortestJobIndex].burst\_time--;

currentTime++;

if (processes[shortestJobIndex].burst\_time == 0) {

processes[shortestJobIndex].completion\_time = currentTime;

processes[shortestJobIndex].turnaround\_time = processes[shortestJobIndex].completion\_time - processes[shortestJobIndex].arrival\_time;

processes[shortestJobIndex].waiting\_time = processes[shortestJobIndex].turnaround\_time - processes[shortestJobIndex].burst\_time;

}

int allCompleted = 1;

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

if (processes[i].burst\_time > 0) {

allCompleted = 0;

break;

}

}

if (allCompleted) {

break;

}

}

printf(" %d \n", currentTime);

}

int main() {

int n;

printf("Enter the number of processes: ");

scanf("%d", &n);

struct Process processes[n];

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

processes[i].id = i + 1;

printf("Enter arrival time and burst time for process %d: ", i + 1);

scanf("%d %d", &processes[i].arrival\_time, &processes[i].burst\_time);

}

SJF(processes, n);

printf("\nProcess\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");

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

printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].id, processes[i].arrival\_time,

processes[i].burst\_time, processes[i].completion\_time, processes[i].turnaround\_time,

processes[i].waiting\_time);

}

return 0;

}