#include<stdio.h>

struct Process

{

int pid;

int burstTime;

int arrivalTime;

int remainingTime;

int waitingTime;

int turnaroundTime;

};

void displayGanttChart(struct Process p[], int n)

{

int i, j;

int total\_time = 0;

printf("\nGantt Chart:\n");

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

{

total\_time += p[i].burstTime;

}

printf(" ");

for(i=0; i<total\_time; i++)

printf("-");

printf("\n|");

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

{

for(j=0; j<p[i].burstTime; j++)

printf(" ");

printf("P%d", p[i].pid);

for(j=0; j<p[i].burstTime; j++)

printf(" ");

printf("|");

}

printf("\n ");

for(i=0; i<total\_time; i++)

printf("-");

printf("\n");

printf("0");

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

{

for(j=0; j<p[i].burstTime; j++)

printf(" ");

printf(" ");

printf("%d", p[i].turnaroundTime);

}

printf("\n\n");

}

void preemptiveSJF(struct Process p[], int n)

{

int i, time, shortest, nextProcess, finishTime;

int remainingTime[n], timeInSeconds = 0, completed = 0;

float totalWaitingTime = 0, totalTurnaroundTime = 0;

// Initializing remaining burst time

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

remainingTime[i] = p[i].burstTime;

while(completed != n)

{

shortest = -1;

// Find process with shortest remaining time

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

{

if(p[i].arrivalTime <= timeInSeconds && remainingTime[i] > 0)

{

if(shortest == -1 || remainingTime[i] < remainingTime[shortest])

shortest = i;

}

}

if(shortest == -1)

{

timeInSeconds++;

continue;

}

// Calculate finish time of current process

finishTime = timeInSeconds + 1;

remainingTime[shortest]--;

if(remainingTime[shortest] == 0)

{

completed++;

p[shortest].waitingTime = finishTime - p[shortest].burstTime - p[shortest].arrivalTime;

p[shortest].turnaroundTime = finishTime - p[shortest].arrivalTime;

totalWaitingTime += p[shortest].waitingTime;

totalTurnaroundTime += p[shortest].turnaroundTime;

}

timeInSeconds++;

}

printf("\nPreemptive Shortest Job First Scheduling:\n");

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

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

{

printf("P%d\t%d\t\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burstTime, p[i].arrivalTime, p[i].waitingTime, p[i].turnaroundTime);

}

printf("\nAverage Waiting Time: %.2f\n", totalWaitingTime/n);

printf("Average Turnaround Time: %.2f\n", totalTurnaroundTime/n);

displayGanttChart(p, n);

}

int main()

{

int n, i;

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

scanf("%d", &n);

struct Process p[n];

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

{

printf("\nEnter the arrival time of P%d: ", i+1);

scanf("%d", &p[i].arrivalTime);

printf("Enter the burst time of P%d: ", i+1);

scanf("%d", &p[i].burstTime);

p[i].pid = i+1;

p[i].remainingTime = p[i].burstTime;

p[i].waitingTime = 0;

p[i].turnaroundTime = 0;

}

preemptiveSJF(p, n);

return 0;

}