#include <iostream>

#include <algorithm>

#include <iomanip>

#include <string.h>

using namespace std;

struct process {

int pid;

int arrival\_time;

int burst\_time;

int start\_time;

int completion\_time;

int turnaround\_time;

int waiting\_time;

int response\_time;

};

int main() {

int n;

struct process p[100];

float avg\_turnaround\_time;

float avg\_waiting\_time;

float avg\_response\_time;

float cpu\_utilisation;

int total\_turnaround\_time = 0;

int total\_waiting\_time = 0;

int total\_response\_time = 0;

int total\_idle\_time = 0;

float throughput;

int burst\_remaining[100];

int is\_completed[100];

memset(is\_completed,0,sizeof(is\_completed));

cout << setprecision(2) << fixed;

cout<<"Enter the number of processes: ";

cin>>n;

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

cout<<"Enter arrival time of process "<<i+1<<": ";

cin>>p[i].arrival\_time;

cout<<"Enter burst time of process "<<i+1<<": ";

cin>>p[i].burst\_time;

p[i].pid = i+1;

burst\_remaining[i] = p[i].burst\_time;

cout<<endl;

}

int current\_time = 0;

int completed = 0;

int prev = 0;

while(completed != n) {

int idx = -1;

int mn = 10000000;

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

if(p[i].arrival\_time <= current\_time && is\_completed[i] == 0) {

if(burst\_remaining[i] < mn) {

mn = burst\_remaining[i];

idx = i;

}

if(burst\_remaining[i] == mn) {

if(p[i].arrival\_time < p[idx].arrival\_time) {

mn = burst\_remaining[i];

idx = i;

}

}

}

}

if(idx != -1) {

if(burst\_remaining[idx] == p[idx].burst\_time) {

p[idx].start\_time = current\_time;

total\_idle\_time += p[idx].start\_time - prev;

}

burst\_remaining[idx] -= 1;

current\_time++;

prev = current\_time;

if(burst\_remaining[idx] == 0) {

p[idx].completion\_time = current\_time;

p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;

p[idx].waiting\_time = p[idx].turnaround\_time - p[idx].burst\_time;

p[idx].response\_time = p[idx].start\_time - p[idx].arrival\_time;

total\_turnaround\_time += p[idx].turnaround\_time;

total\_waiting\_time += p[idx].waiting\_time;

total\_response\_time += p[idx].response\_time;

is\_completed[idx] = 1;

completed++;

}

}

else {

current\_time++;

}

}

int min\_arrival\_time = 10000000;

int max\_completion\_time = -1;

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

min\_arrival\_time = min(min\_arrival\_time,p[i].arrival\_time);

max\_completion\_time = max(max\_completion\_time,p[i].completion\_time);

}

avg\_turnaround\_time = (float) total\_turnaround\_time / n;

avg\_waiting\_time = (float) total\_waiting\_time / n;

avg\_response\_time = (float) total\_response\_time / n;

cpu\_utilisation = ((max\_completion\_time - total\_idle\_time) / (float) max\_completion\_time )\*100;

throughput = float(n) / (max\_completion\_time - min\_arrival\_time);

cout<<endl<<endl;

cout<<"#P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"RT\t"<<"\n"<<endl;

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

cout<<p[i].pid<<"\t"<<p[i].arrival\_time<<"\t"<<p[i].burst\_time<<"\t"<<p[i].start\_time<<"\t"<<p[i].completion\_time<<"\t"<<p[i].turnaround\_time<<"\t"<<p[i].waiting\_time<<"\t"<<p[i].response\_time<<"\t"<<"\n"<<endl;

}

cout<<"Average Turnaround Time = "<<avg\_turnaround\_time<<endl;

cout<<"Average Waiting Time = "<<avg\_waiting\_time<<endl;

cout<<"Average Response Time = "<<avg\_response\_time<<endl;

cout<<"CPU Utilization = "<<cpu\_utilisation<<"%"<<endl;

cout<<"Throughput = "<<throughput<<" process/unit time"<<endl;

}