// round robin method

#include<stdio.h>

#include<stdlib.h>

struct queue{

int pid;

struct queue\* next;

};

struct queue\* rq = NULL;

struct queue\* create(int p){

struct queue\* nn = malloc(sizeof(struct queue));

nn->pid = p;

nn->next = NULL;

return nn;

}

void enqueue(int p){

struct queue \* nn = create(p);

if(rq==NULL)

rq=nn;

else{

struct queue\* temp = rq;

while(temp->next!=NULL)

temp = temp->next;

temp->next = nn;

}

}

int dequeue(){

int x=0;

if (rq == NULL)

return x;

else{

struct queue\* temp = rq;

x = temp->pid;

rq = rq->next;

free(temp);

}

return x;

}

void printq(){

struct queue\* temp = rq;

while(temp!=NULL){

printf("%d\t",temp->pid);

temp = temp->next;

}

printf("\n");

}

void swap(int \*a,int \*b){

\*a = \*a + \*b;

\*b = \*a - \*b;

\*a = \*a - \*b;

}

void sort(int \*pid, int \*at, int \*bt, int n){

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

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

if(at[i]<at[j]){

swap(&at[i],&at[j]);

swap(&bt[i],&bt[j]);

swap(&pid[i],&pid[j]);

}

}

}

}

int main(){

int n,t,x=1;

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

scanf("%d",&n);

printf("Enter the time quantum : ");

scanf("%d",&t);

int pid[n],at[n],bt1[n],ct[n],tat[n],wt[n],bt2[n],rt[n];

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

printf("Enter arrival time and burst time : ");

scanf("%d%d",&at[i],&bt1[i]);

pid[i]=i+1;

}

sort(pid,at,bt1,n);

enqueue(pid[0]);

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

bt2[i]=bt1[i];

rt[i] = -1;

}

int count = 0;

int ctvar = at[0];

while (count != n){

int curp = rq->pid;

int curi = 0;

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

if(pid[i] == curp){

curi = i;

break;

}

}

if(rt[curi]==-1){

rt[curi] = ctvar - at[curi];

}

if(bt2[curi]<=t){

ctvar += bt2[curi];

bt2[curi] = 0;

}

else{

ctvar += t;

bt2[curi] -=t;

}

while(at[x]<=ctvar && x<n){

enqueue(pid[x]);

x +=1;

}

if(bt2[curi]>0)

enqueue(pid[curi]);

if(bt2[curi] == 0){

count +=1;

ct[curi] = ctvar;

}

dequeue();

}

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

tat[i]=ct[i]-at[i];

wt[i]=tat[i]-bt1[i];

}

float avg\_tat=0;

float avg\_wt=0;

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

avg\_tat+=tat[i];

avg\_wt+=wt[i];

}

printf("pid\tat\tbt\tct\ttat\twt\trt\n\n");

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

printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t",pid[i],at[i],bt1[i],ct[i],tat[i],wt[i],rt[i]);

printf("\n");

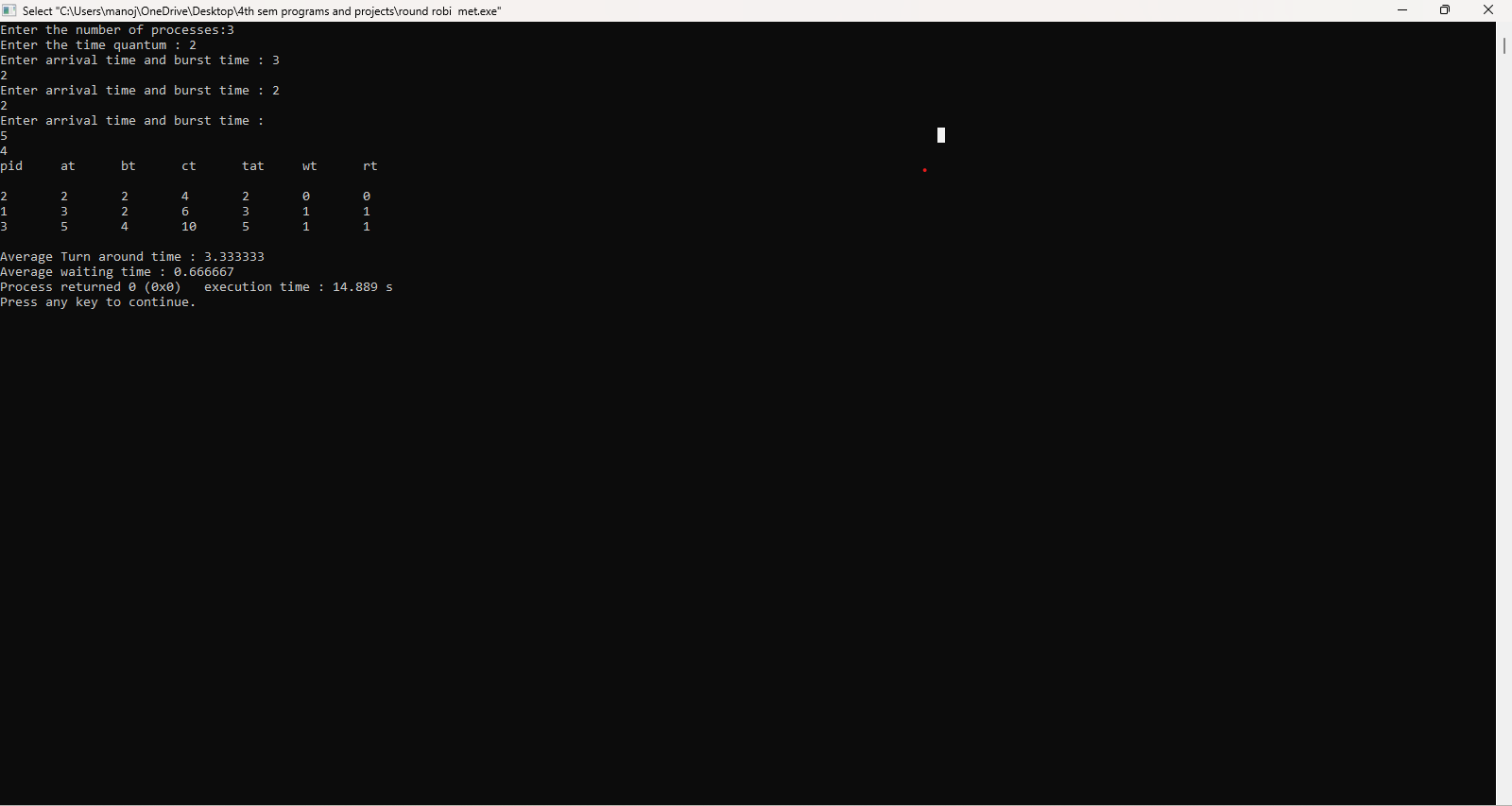
}

printf("\nAverage Turn around time : %f",avg\_tat/n);

printf("\nAverage waiting time : %f",avg\_wt/n);

return 0;

}



//priority

#include<stdio.h>

#include<stdlib.h>

void swap(int \*a,int \*b){

\*a = \*a + \*b;

\*b = \*a - \*b;

\*a = \*a - \*b;

}

void sort(int \*pid, int \*at, int \*bt, int \*prior, int n){

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

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

if(at[i]<at[j]){

swap(&at[i],&at[j]);

swap(&bt[i],&bt[j]);

swap(&pid[i],&pid[j]);

swap(&prior[i],&prior[j]);

}

}

}

}

int highest\_priority(int \*prior, int s, int e){

int x = prior[s];

int j = s;

for(int i=s;i<e;i++){

if (prior[i]>x){

x = prior[i];

j = i;

}

}

return j;

}

int main(){

int n,t,x;

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

scanf("%d",&n);

int pid[n],at[n],bt1[n],ct[n],tat[n],wt[n],bt2[n],rt[n],prior[n];

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

printf("Enter arrival time, burst time and priority : ");

scanf("%d%d%d",&at[i],&bt1[i],&prior[i]);

pid[i]=i+1;

}

sort(pid,at,bt1,prior,n);

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

bt2[i]=bt1[i];

rt[i] = -1;

}

int arvc = 0;

int count = 0;

int ctvar = at[0];

int curi = 0;

while (count != n){

if(rt[curi]==-1){

rt[curi] = ctvar - at[curi];

}

if(arvc == n){

ctvar += bt2[curi];

bt2[curi] = 0;

}

else{

ctvar += 1;

bt2[curi] -= 1;

}

for(int i = 0;at[i]<=ctvar;i++){

arvc +=1;

x = i;

}

if(bt2[curi] == 0){

count +=1;

ct[curi] = ctvar;

prior[curi] = -1;

}

curi = highest\_priority(prior,0,x+1);

}

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

tat[i]=ct[i]-at[i];

wt[i]=tat[i]-bt1[i];

}

float avg\_tat=0;

float avg\_wt=0;

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

avg\_tat+=tat[i];

avg\_wt+=wt[i];

}

printf("pid\tat\tbt\tct\ttat\twt\trt\n\n");

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

printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t",pid[i],at[i],bt1[i],ct[i],tat[i],wt[i],rt[i]);

printf("\n");

}

printf("\nAverage Turn around time : %f",avg\_tat/n);

printf("\nAverage waiting time : %f",avg\_wt/n);

return 0;

}

