#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include <stdbool.h>

struct Process

{

char pname[10];

int arr\_time;

int burst\_time;

int priority;

int wt,tat;

};

void Input\_Process(struct Process\* P ,int n,bool priority);

void FCFS(struct Process\* P ,int n);

void Output\_Process(struct Process\* P ,int n);

int main()

{

bool flag=false;

int n,choice;

printf("Enter number of processes:");

scanf("%d",&n);

struct Process\* P = (struct Process\*)malloc(n\*sizeof(struct Process));

FCFS(P,n);

Output\_Process(P,n);

}

void Input\_Process(struct Process\* P ,int n,bool priority)

{

int i,j;

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

{

fflush(stdin);

printf("\nEnter %d process name:",i+1);

scanf("%s",P[i].pname);

printf("Enter %d arrival time:",i+1);

scanf("%d",&P[i].arr\_time);

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

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

if(priority)

{

printf("Enter %d priority:",i+1);

scanf("%d",&P[i].priority);

}

else

P[i].priority = 1;

}

//to sort by arrival time

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

{

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

{

if(P[j].arr\_time > P[j+1].arr\_time)

{

struct Process temp = P[j];

P[j] = P[j+1];

P[j+1]=temp;

}

}

}

}

void FCFS(struct Process\* P ,int n)

{

Input\_Process(P,n,false);

int t = 0,i;

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

{

if(t < P[i].arr\_time)

t = P[i].arr\_time;

P[i].wt = t-P[i].arr\_time;

P[i].tat = P[i].burst\_time+P[i].wt;

t = t+P[i].burst\_time;

}

}

void Output\_Process(struct Process\* P ,int n)

{

int i;

printf("\nNAME\t\tARRIVAL\_TIME\t\tBURST\_TIME\t\tPRIORITY\tWAITING\_TIME\tTURN\_AROUND\_TIME\n");

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

printf("%s\t\t%d\t\t\t%d\t\t\t%d\t\t%d\t\t%d\n",P[i].pname,P[i].arr\_time,P[i].burst\_time,P[i].priority,P[i].wt,P[i].tat);

double wt=0,tat=0;

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

{

wt += P[i].wt;

tat += P[i].tat;

}

wt /= n;

tat /= n;

printf("\nAverage:\nWaiting Time = %f\nTurn Around Time = %f",wt,tat);

}

ALGORITHM FCFS(N)

[N -> number of processes]

STEP 1: Declare P[N] where P is an array of Process Object which contains a name, arrival\_time, burst\_time, waiting\_time and turn\_around\_time.

STEP 2: Input values of name, burst time and arrival of each process from the user.

STEP 3: Sort the Process according to their arrival time.

STEP 4: temp := 0;

    for i := 1 to n do

if temp < P[i]->arrival\_time then

temp := P[i]->arrival\_time

                end if.

P[i]->waiting\_time := temp - P[i]->arrrival\_time.

P[i]->turn\_around\_time := P[i]->burst\_time + P[i]->waiting\_time.

temp : = temp + P[i]->burst\_time.

            end for.

STEP 5: avg\_waiting\_time :=0,avg\_turn\_around\_time :=0.

for i :=1 to N do

avg\_waiting\_time  := avg\_waiting\_time + P[i]->waiting\_timet

avg\_turn\_around\_time := avg\_turn\_around\_time + P[i]->turn\_around\_time

        end for.

avg\_waiting\_time  := avg\_waiting\_time /N;

avg\_turn\_around\_time  := avg\_turn\_around\_time /N;

STEP 6: Exit.