package pakage;

import java.util.\*;

public class RoundRobin {

private static Scanner inp = new Scanner(System.in);

//Driver Code

public static void main(String[] args){

int n,tq, timer = 0, maxProccessIndex = 0;

float avgWait = 0, avgTT = 0;

System.out.print("\nEnter the time quantum : ");

tq = inp.nextInt();

System.out.print("\nEnter the number of processes : ");

n = inp.nextInt();

int arrival[] = new int[n];

int burst[] = new int[n];

int wait[] = new int[n];

int turn[] = new int[n];

int queue[] = new int[n];

int temp\_burst[] = new int[n];

boolean complete[] = new boolean[n];

System.out.print("\nEnter the arrival time of the processes : ");

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

arrival[i] = inp.nextInt();

System.out.print("\nEnter the burst time of the processes : ");

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

burst[i] = inp.nextInt();

temp\_burst[i] = burst[i];

}

for(int i = 0; i < n; i++){ //Initializing the queue and complete array

complete[i] = false;

queue[i] = 0;

}

while(timer < arrival[0]) //Incrementing Timer until the first process arrives

timer++;

queue[0] = 1;

while(true){

boolean flag = true;

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

if(temp\_burst[i] != 0){

flag = false;

break;

}

}

if(flag)

break;

for(int i = 0; (i < n) && (queue[i] != 0); i++){

int ctr = 0;

while((ctr < tq) && (temp\_burst[queue[0]-1] > 0)){

temp\_burst[queue[0]-1] -= 1;

timer += 1;

ctr++;

//Updating the ready queue until all the processes arrive

checkNewArrival(timer, arrival, n, maxProccessIndex, queue);

}

if((temp\_burst[queue[0]-1] == 0) && (complete[queue[0]-1] == false)){

turn[queue[0]-1] = timer; //turn currently stores exit times

complete[queue[0]-1] = true;

}

//checks whether or not CPU is idle

boolean idle = true;

if(queue[n-1] == 0){

for(int k = 0; k < n && queue[k] != 0; k++){

if(complete[queue[k]-1] == false){

idle = false;

}

}

}

else

idle = false;

if(idle){

timer++;

checkNewArrival(timer, arrival, n, maxProccessIndex, queue);

}

//Maintaining the entries of processes after each premption in the ready Queue

queueMaintainence(queue,n);

}

}

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

turn[i] = turn[i] - arrival[i];

wait[i] = turn[i] - burst[i];

}

System.out.print("\nProgram No.\tArrival Time\tBurst Time\tWait Time\tTurnAround Time" + "\n");

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

System.out.print(i+1+"\t\t"+arrival[i]+"\t\t"+burst[i]

+"\t\t"+wait[i]+"\t\t"+turn[i]+ "\n");

}

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

avgWait += wait[i];

avgTT += turn[i];

}

System.out.print("\nAverage wait time : "+(avgWait/n)

+"\nAverage Turn Around Time : "+(avgTT/n));

}

public static void queueUpdation(int queue[],int timer,int arrival[],int n, int maxProccessIndex){

int zeroIndex = -1;

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

if(queue[i] == 0){

zeroIndex = i;

break;

}

}

if(zeroIndex == -1)

return;

queue[zeroIndex] = maxProccessIndex + 1;

}

public static void checkNewArrival(int timer, int arrival[], int n, int maxProccessIndex,int queue[]){

if(timer <= arrival[n-1]){

boolean newArrival = false;

for(int j = (maxProccessIndex+1); j < n; j++){

if(arrival[j] <= timer){

if(maxProccessIndex < j){

maxProccessIndex = j;

newArrival = true;

}

}

}

if(newArrival) //adds the index of the arriving process(if any)

queueUpdation(queue,timer,arrival,n, maxProccessIndex);

}

}

public static void queueMaintainence(int queue[], int n){

for(int i = 0; (i < n-1) && (queue[i+1] != 0) ; i++){

int temp = queue[i];

queue[i] = queue[i+1];

queue[i+1] = temp;

}

}

}