**TO RUN LINUX COMMANDS** SOFTWARE REQUIRED: Linux **PROGRAM:**

~$ mkdir home

~$ mkdir IPEC

~$ mkdir AKG

~$ cd IPEC

~/IPEC$ mkdir cs

~/IPEC$ mkdir me

~/IPEC$ mkdir it

~/IPEC$ cd

~$ cd AKG

~/AKG$ mkdir me

~/AKG$ mkdir ec

~/AKG$ cd

~$ cat >file sec A

sec B sec C sec D

~$ cat >sections sec E

sec F

~$ cat >branch CSE

CSE CSE CSE IT

IT

~$ cat file sections branch>fil~$ cat file2 sec A

sec B sec C sec D sec E sec F

CSE CSE CSE CSE IT

IT

~$ rm sections

~$ cat sections

cat: sections: No such file or directory

~$ ls

2022-05-12-112945.term AKG IPEC branch file file2 home

~$ cp branch branchc

~$ ls

2022-05-12-112945.term AKG IPEC branch branchc file file2 home

~$ cat branchc CSE

CSE CSE CSE IT

IT

~$ vi file2

~$ cat file2 sec A

sec B sec C sec D sec E sec F CSE CSE CSE CSE IT

IT ME ME

~$ ls

2022-05-12-112945.term AKG IPEC branch branchc file file2 home

~$ mv file2 class

~$ ls

2022-05-12-112945.term AKG IPEC branch branchc class file home

~$ cat >happy

Happiness is the key towards leading a happy life. Happiness has no common definition and meaning that is accepted by all.

Every individual defines happiness in his/ her own unique way the way he/she feels it.Happiness is the key towards leading a happy life. Happiness has no common definition and meaning that is accepted by all.

Every individual defines happiness in his/ her own unique way the way he/she feels it.It is also said that happiness is attaining something which is most important to you as compared to all other things. Happiness is also said to be a way to have positive attitude towards life. Happiness can also be attained by doing good deeds towards others or treating others the way that makes you and them happy as well.Happiness is finding reasons behind your existence. The extent of happiness cannot be measured by any scale. Happiness is neither tradable nor profitable. Everybody wants happiness in their lives. ~$ grep -i "Happiness" happy

Happiness is the key towards leading a happy life. Happiness has no common definition and meaning that is accepted by all.

Every individual defines happiness in his/ her own unique way the way he/she feels it.Happiness is the key towards leading a happy life. Happiness has no common definition and meaning that is accepted by all.

Every individual defines happiness in his/ her own unique way the way he/she feels it.It is also said that happiness is attaining something which is most important to you as compared to all other things. Happiness is also said to be a way to have positive attitude towards life. Happiness can also be attained by doing good deeds towards others or treating others the way that makes you and them happy as well.Happiness is finding reasons behind your existence. The extent of happiness cannot be measured by any scale. Happiness is neither tradable nor profitable. Everybody wants happiness in their lives.

~$ grep -c "happiness" happy 2

~$ grep -c "Happiness" happy 3

~$ grep -n "Happiness" happy

1:Happiness is the key towards leading a happy life. Happiness has no common definition and meaning that is accepted by all.

3:Every individual defines happiness in his/ her own unique way the way he/she feels it.Happiness is the key towards leading a happy life. Happiness has no common definition and meaning that is accepted by all.

5:Every individual defines happiness in his/ her own unique way the way he/she feels it.It is also said that happiness is attaining something which is most important to you as compared to all other things. Happiness is also said to be a way to have positive attitude towards life. Happiness can also be attained by doing good deeds towards others or treating others the way that makes you and them happy as well.Happiness is finding reasons behind your existence. The extent of happiness cannot be measured by any scale. Happiness is neither tradable nor profitable. Everybody wants happiness in their lives.

~$ grep -v "Happiness" happy

# WRITE A PROGRAM FOR FCFS WITH 0 ARRIVAL FOR ALL PROCESSES

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> void main()

{

int i,j,n,bt[50],wt[50],tat[50]; float awt=0,avtat=0;

printf("Enter total number of processes : "); scanf("%d",&n);

printf("Enter the Burst Time of Process : \n"); for(i=0;i<n;i++)

{

printf("P[%d]:",i+1);

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

}

printf("Process\t Burst Time\t Waiting Time\t Turn Around Time\n"); for(i=0;i<n;i++)

{

wt[i]=0;

tat[i]=0;

//Calculating the Waiting Time for(j=0;j<i;j++)

{

wt[i]=wt[i]+bt[j];

}

//Calculating the Turn Around Time tat[i]=wt[i]+bt[i];

//Calculating the Total Waiting Time awt=awt+wt[i];

//Calculating the Total Turn Around Time avtat=avtat+tat[i]; printf("%d\t%d\t\t\t%d\t\t%d\n",i+1,bt[i],wt[i],tat[i]);

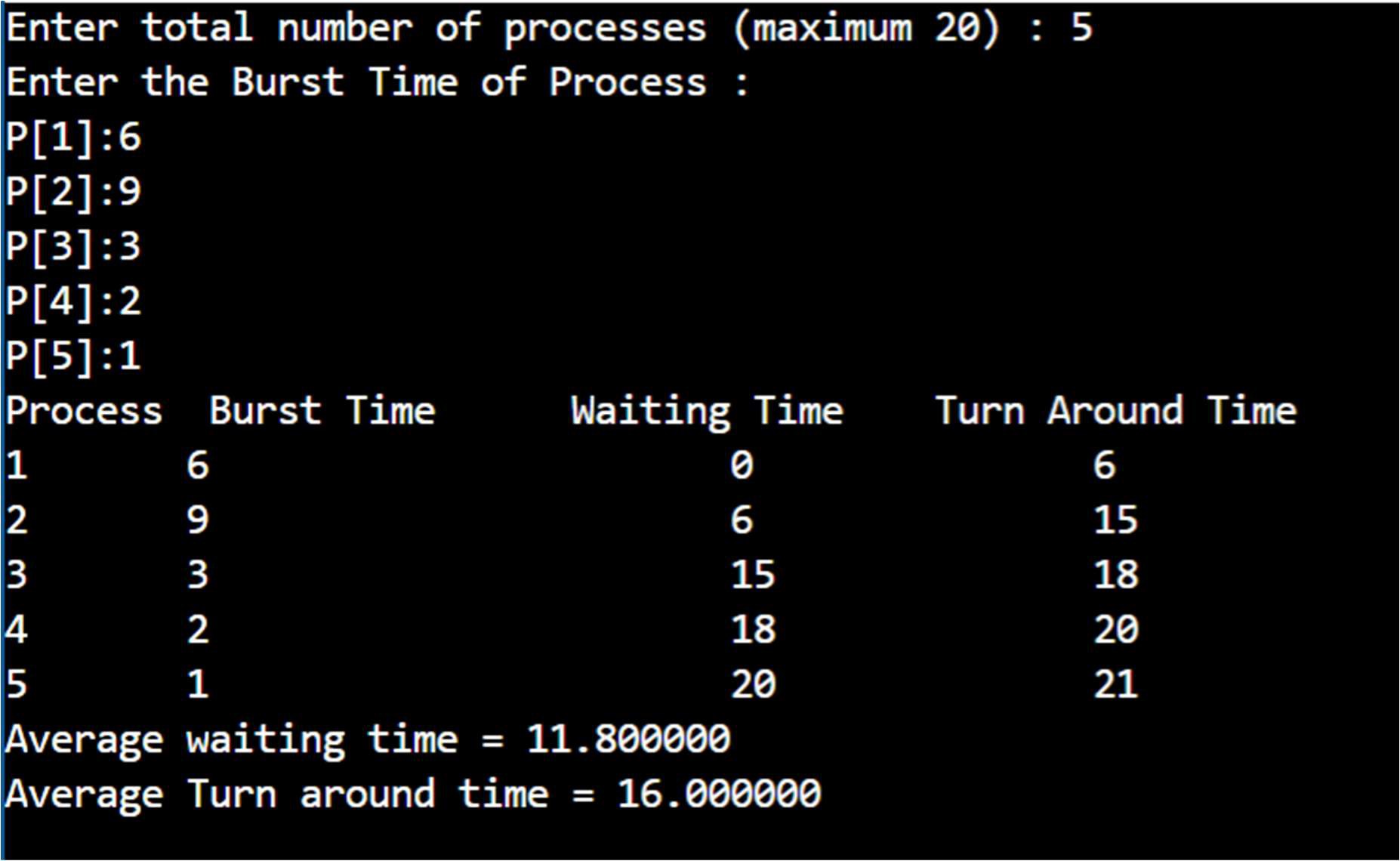
}

//Printing the Average Turn Around time and Waiting Time printf("Average waiting time = %f\n",awt/n);

printf("Average Turn around time = %f",avtat/n);

}

OUTPUT:



# WRITE A PROGRAM FOR FCFS WITH DIFFERENT ARRIVAL FOR ALL PROCESSES

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> int main()

{

int bt[10]={0},at[10]={0},tat[10]={0},wt[10]={0},ct[10]={0};

int n,sum=0;

float totalTAT=0,totalWT=0;

printf("Enter total number of processes : "); scanf("%d",&n);

printf("Enter Arrival Time and Burst Time for each process : \n"); for(int i=0;i<n;i++)

{

printf("Arrival time of process P[%d] : ",i+1); scanf("%d",&at[i]);

printf("Burst time of process P[%d] : ",i+1); scanf("%d",&bt[i]);

printf("\n");

}

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

{

sum+=bt[j]; ct[j]+=sum;

}

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

{

tat[k]=ct[k]-at[k]; totalTAT+=tat[k];

}

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

{

wt[k]=tat[k]-bt[k]; totalWT+=wt[k];

}

printf("\nProcess\tArrival Time\tBurst Time\tCompletion Time\tTurn Around Time\tWaiting Time ");

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

{

printf("P[%d]\t %d\t %d\t %d\t %d\t %d\n",i+1,at[i],bt[i],ct[i],tat[i],wt[i]);

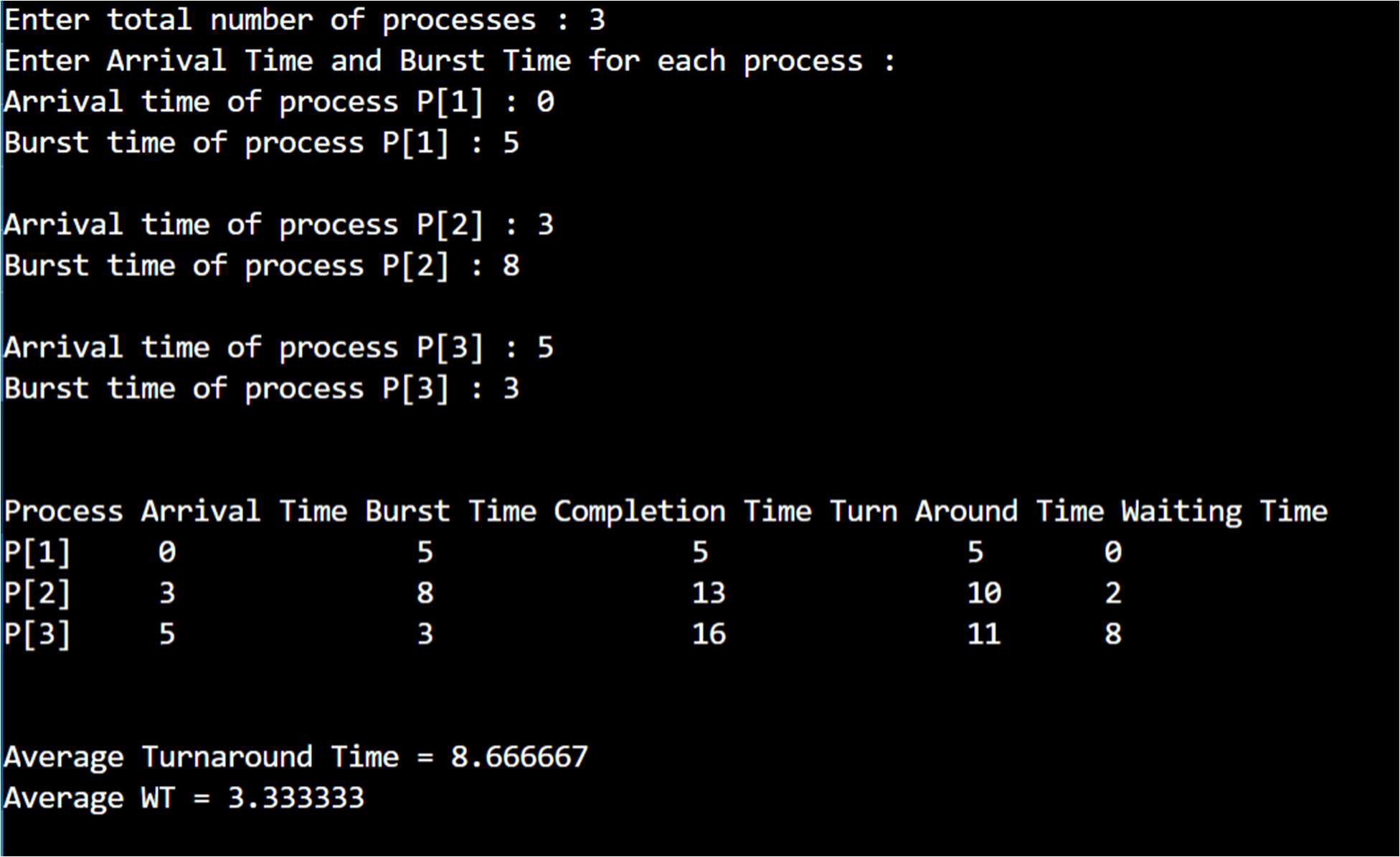
}

printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n); printf("Average WT = %f\n\n",totalWT/n);

return 0;

}

**OUTPUT:**



# WRITE A PROGRAM FOR SJF NON PRE-EMPTIVE

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> int main()

{

int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp; float avg\_wt,avg\_tat;

printf("Enter total number of processes : "); scanf("%d",&n);

printf("\nEnter Burst Time of process : \n"); for(i=0;i<n;i++)

{

printf("P[%d] : ",i+1);

scanf("%d",&bt[i]); p[i]=i+1;

}

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

{

pos=i; for(j=i+1;j<n;j++)

{

if(bt[j]<bt[pos]) pos=j;

}

temp=bt[i]; bt[i]=bt[pos]; bt[pos]=temp; temp=p[i]; p[i]=p[pos]; p[pos]=temp;

}

wt[0]=0;

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

{

wt[i]=0; for(j=0;j<i;j++)

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=(float)total/n; total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time"); for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i]; total+=tat[i];

printf("\n%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

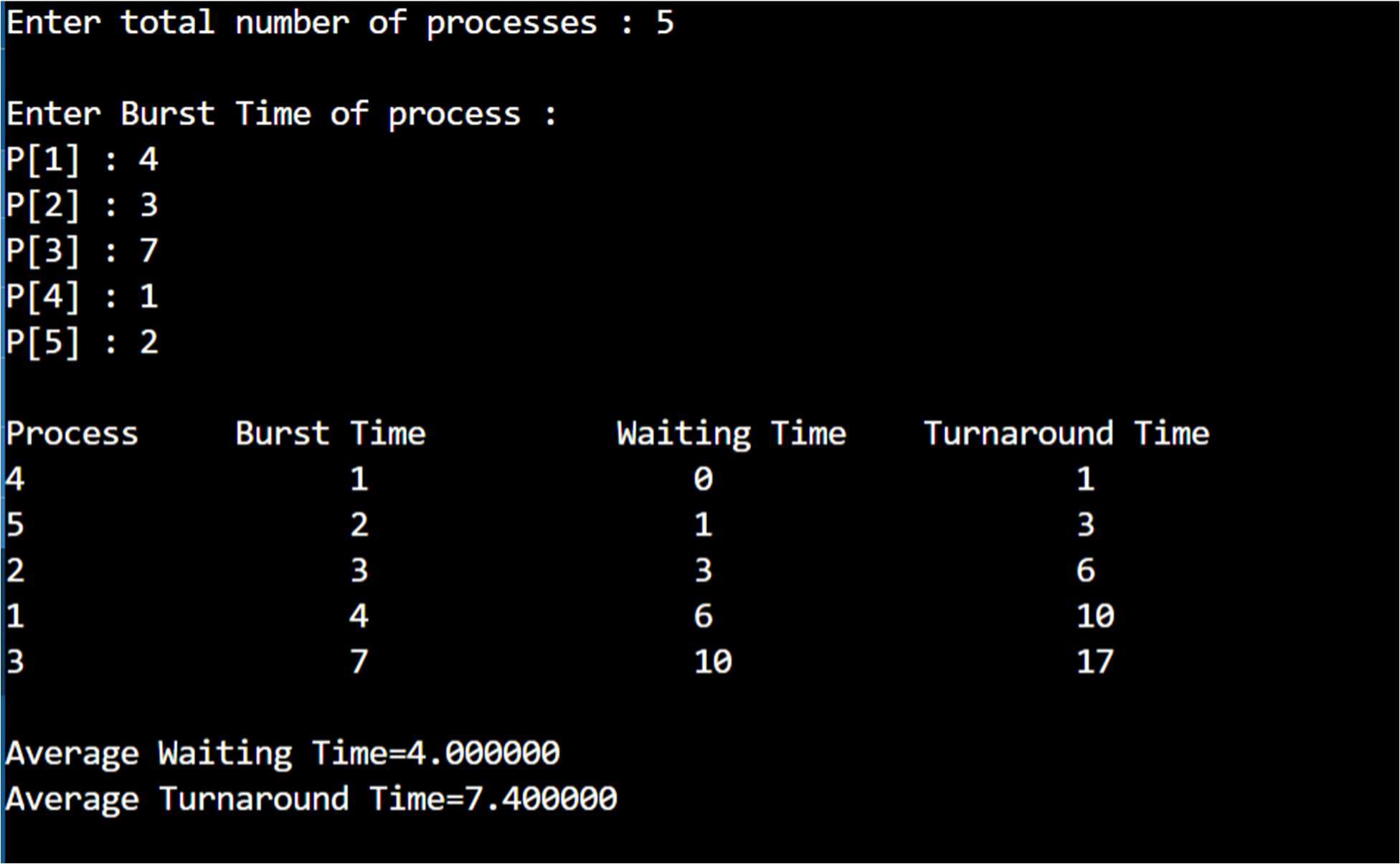
}

avg\_tat=(float)total/n;

printf("\n\nAverage Waiting Time=%f",avg\_wt); printf("\nAverage Turnaround Time=%f\n",avg\_tat);

}

**OUTPUT:**



# WRITE A PROGRAM FOR SJF PRE-EMPTIVE

SOFTWARE REQUIRED: C

## PROGRAM:

#include <stdio.h> int main()

{

int arrival\_time[10], burst\_time[10], temp[10]; int i, smallest, count = 0, time,n;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time; printf("\nEnter the total number of Processes : "); scanf("%d", &n);

printf("Enter the Burst Time and Arrival Time of the Process : \n"); for(i = 0; i < n; i++)

{

printf("Arrival time of process[%d] : ",i+1); scanf("%d",&arrival\_time[i]);

printf("Burst time of process[%d] : ",i+1); scanf("%d",&burst\_time[i]);

printf("\n"); temp[i] = burst\_time[i];

}

burst\_time[9] = 9999;

for(time = 0; count != n; time++)

{

smallest = 9;

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

{

> 0)

}

if(arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i]

{

smallest = i;

}

burst\_time[smallest]--; if(burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest]; turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

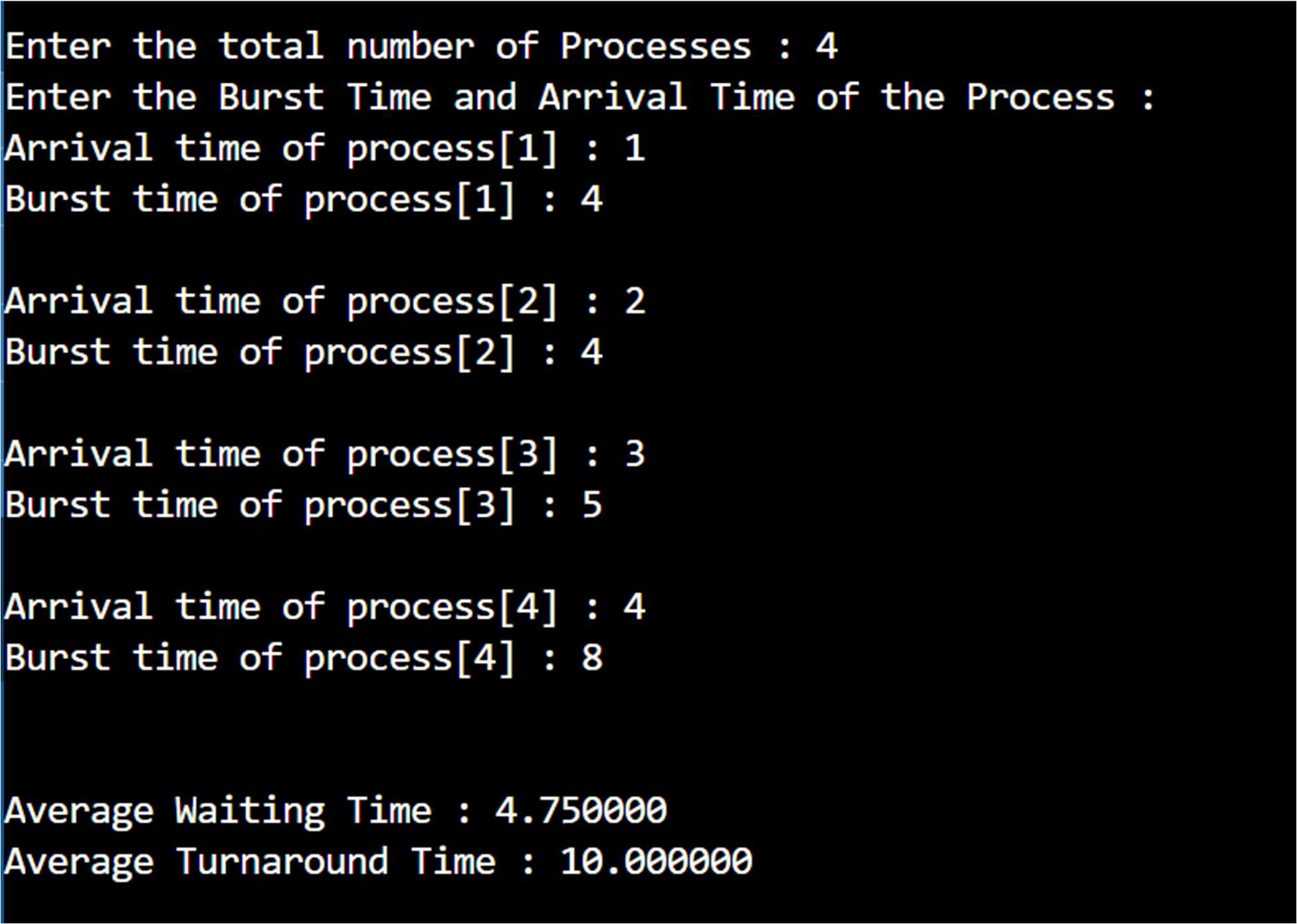
}

average\_waiting\_time = wait\_time / n; average\_turnaround\_time = turnaround\_time / n;

printf("\n\nAverage Waiting Time : %lf\n", average\_waiting\_time); printf("Average Turnaround Time : %lf\n", average\_turnaround\_time); return 0;

}

**OUTPUT:**



# WRITE A PROGRAM FOR NON PRE-EMPTIVE PRIORITY SCHEDULING

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> int main()

{

int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp; float avg\_wt,avg\_tat;

printf("Enter Total Number of Process : "); scanf("%d",&n);

printf("\nEnter Burst Time and Priority : \n"); for(i=0;i<n;i++)

{

printf("\nP[%d]\n",i+1); printf("Burst Time : "); scanf("%d",&bt[i]); printf("Priority : "); scanf("%d",&pr[i]);

p[i]=i+1; //contains process number

}

//sorting burst time, priority and process number in ascending order using selection sort

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

{

pos=i; for(j=i+1;j<n;j++)

{

if(pr[j]<pr[pos]) pos=j;

}

temp=pr[i]; pr[i]=pr[pos]; pr[pos]=temp; temp=bt[i]; bt[i]=bt[pos]; bt[pos]=temp; temp=p[i]; p[i]=p[pos]; p[pos]=temp;

}

wt[0]=0; //waiting time for first process is zero

//calculate waiting time for(i=1;i<n;i++)

{

wt[i]=0; for(j=0;j<i;j++)

wt[i]+=bt[j]; total+=wt[i];

}

avg\_wt=total/n; //average waiting time total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time"); for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i]; //calculate turnaround time total+=tat[i];

printf("\nP[%d]\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

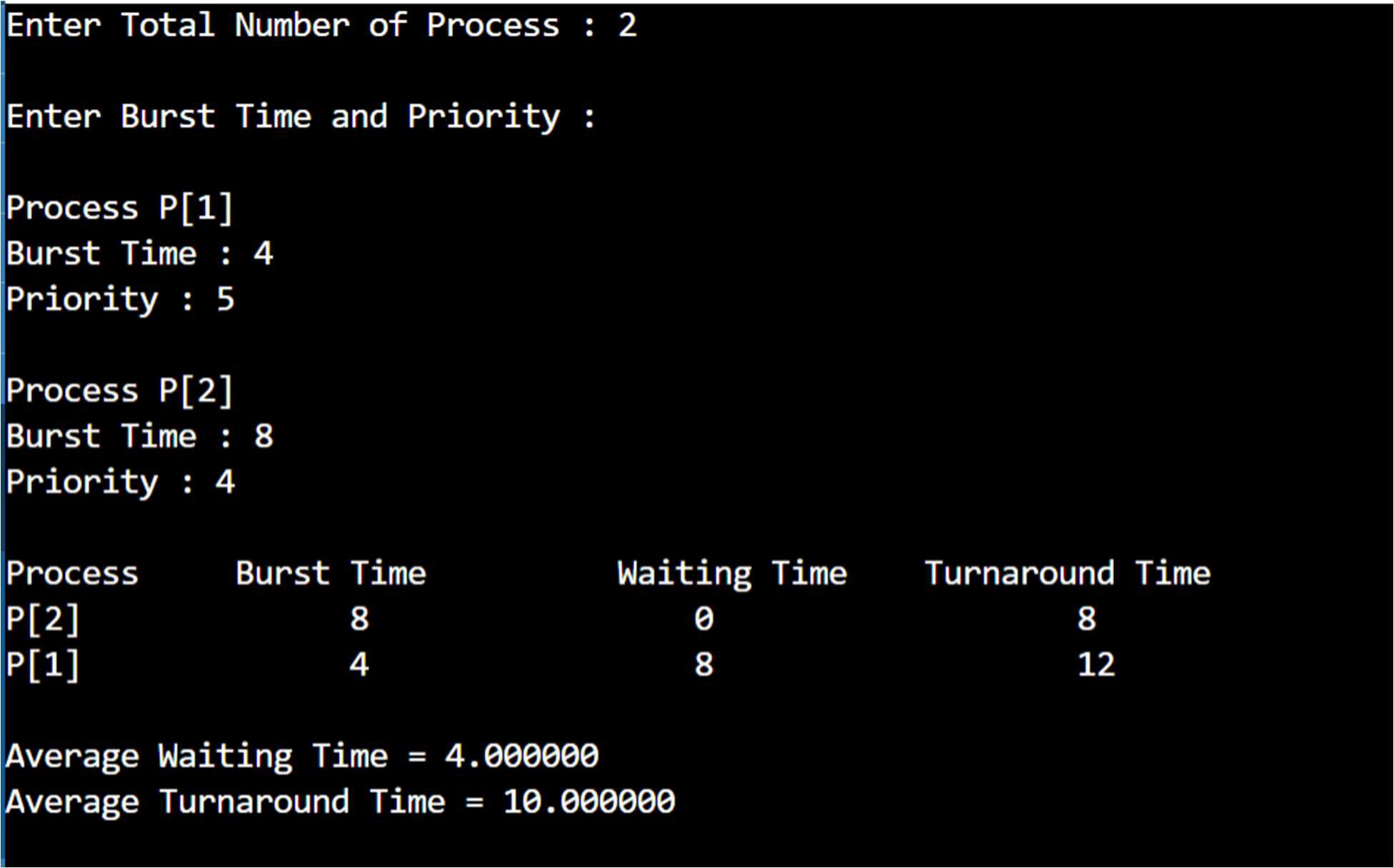
}

avg\_tat=total/n; //average turnaround time printf("\n\nAverage Waiting Time = %f",avg\_wt); printf("\nAverage Turnaround Time = %f\n",avg\_tat);

return 0;

}

**OUTPUT:**



# WRITE A PROGRAM FOR PRE-EMPTIVE PRIORITY SCHEDULING

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> struct process

{

int WT,AT,BT,TAT,PT;

};

struct process a[10]; int main()

{

int n,temp[10],t,count=0,short\_p;

float total\_WT=0,total\_TAT=0,Avg\_WT,Avg\_TAT; printf("\nEnter total number of the processes : "); scanf("%d",&n);

printf("Enter the arrival time , burst time and priority of the process : \n"); for(int i=0;i<n;i++)

{

printf("Process P[%d] : ",i+1); printf("\nArrival Time : "); scanf("%d",&a[i].AT); printf("Burst Time : "); scanf("%d",&a[i].BT); printf("Priority : "); scanf("%d",&a[i].PT); printf("\n");

// copying the burst time in

// a temp array fot futher use temp[i]=a[i].BT;

}

// we initialize the burst time

// of a process with maximum a[9].PT=10000;

for(t=0;count!=n;t++)

{

short\_p=9;

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

{

if(a[short\_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)

{

short\_p=i;

}

}

a[short\_p].BT=a[short\_p].BT-1;

// if any process is completed if(a[short\_p].BT==0)

{

// one process is completed

// so count increases by 1 count++;

a[short\_p].WT=t+1-a[short\_p].AT-temp[short\_p]; a[short\_p].TAT=t+1-a[short\_p].AT;

// total calculation total\_WT=total\_WT+a[short\_p].WT; total\_TAT=total\_TAT+a[short\_p].TAT;

}

}

Avg\_WT=total\_WT/n; Avg\_TAT=total\_TAT/n;

// printing of the answer

printf("Process Waiting Time Turn Around Time\n"); for(int i=0;i<n;i++)

{

printf("%d\t\t%d\t\t%d\n",i+1,a[i].WT,a[i].TAT);

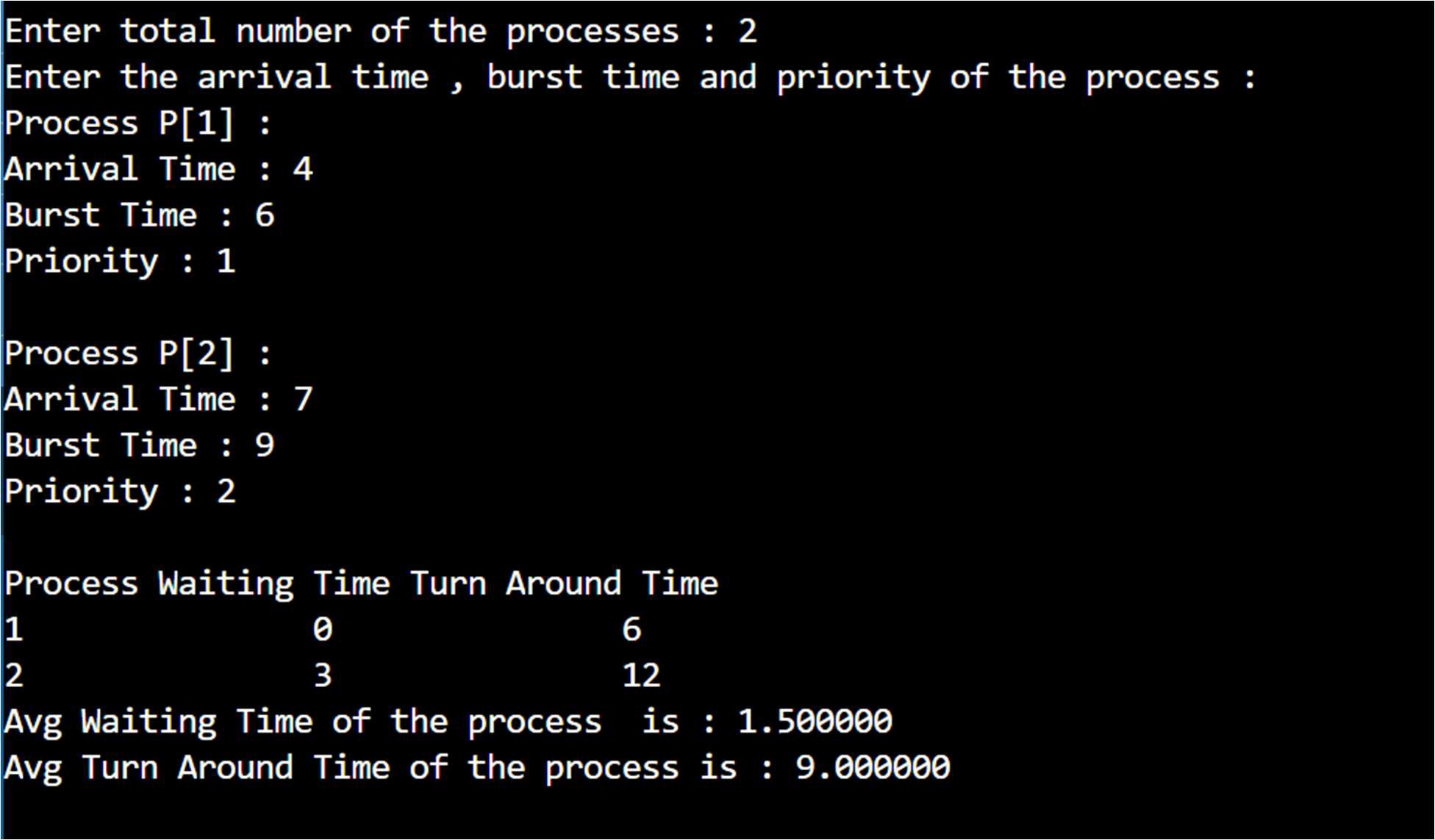
}

printf("Avg Waiting Time of the process is : %f\n",Avg\_WT); printf("Avg Turn Around Time of the process is : %f\n",Avg\_TAT);

return 0;

}

**OUTPUT:**



# WRITE A PROGRAM FOR ROUND ROBIN ALGORITHM

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> #include<conio.h> void main()

{

// initlialize the variable name

int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10]; float avg\_wt, avg\_tat;

printf("Enter total number of processes in the system : "); scanf("%d", &NOP);

y = NOP; // Assign the number of process to variable y

// Use for loop to enter the details of the process like Arrival time and the Burst Time for(i=0; i<NOP; i++)

{

printf("\n Enter the Arrival and Burst time of the P[%d]\n", i+1); printf(" Arrival time is: "); // Accept arrival time

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

printf(" Burst time is: "); // Accept the Burst time scanf("%d", &bt[i]);

temp[i] = bt[i]; // store the burst time in temp array

}

// Accept the Time qunat

printf("Enter the Time Quantum for the process : "); scanf("%d", &quant);

// Display the process No, burst time, Turn Around Time and the waiting time printf("\n Process Burst Time Turn Around Time Waiting Time "); for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0) // define the conditions

{

sum = sum + temp[i]; temp[i] = 0;

count=1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - quant; sum = sum + quant;

}

if(temp[i]==0 && count==1)

{

y--; //decrement the process no.

printf("\n P[%d]\t\t %d \t\t %d \t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]); wt = wt+sum-at[i]-bt[i];

tat = tat+sum-at[i]; count =0;

}

if(i==NOP-1)

{

i=0;

}

else if(at[i+1]<=sum)

{

i++;

}

else

{

i=0;

}

}

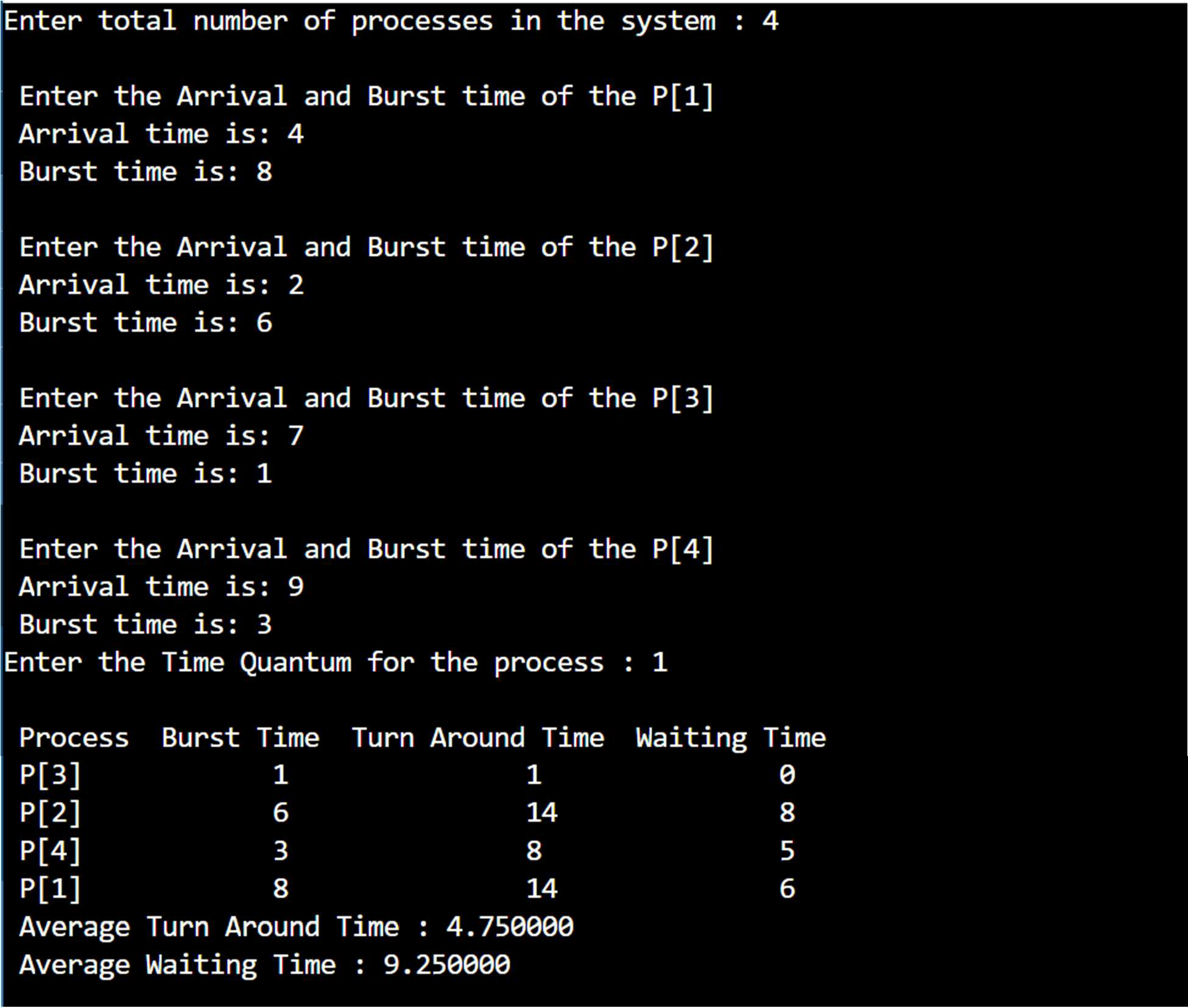
// represents the average waiting time and Turn Around time avg\_wt = wt \* 1.0/NOP;

avg\_tat = tat \* 1.0/NOP;

printf("\n Average Turn Around Time : %f", avg\_wt); printf("\n Average Waiting Time : %f", avg\_tat); getch();

}

**OUTPUT:**



# WRITE A PROGRAM FOR BANKER ALGORITHM

SOFTWARE REQUIRED: C

## PROGRAM:

#include<stdio.h> #include<conio.h> void main() {

int k=0,output[10],d=0,t=0,ins[5],i,avail[5],allocated[10][5],need[10][5],MAX[10][5],pno,P[10]

,j,rz, count=0;

printf("\n Enter total number of resources : "); scanf("%d", &rz);

printf("\n Enter the max instances of each resources : \n"); for (i=0;i<rz;i++) {

avail[i]=0;

printf(" %c = ",(i+97));

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

}

printf("\n Enter the number of processes : "); scanf("%d", &pno);

printf("\n Enter the allocation matrix : \n "); for (i=0;i<rz;i++)

printf(" %c",(i+97)); printf("\n");

for (i=0;i <pno;i++) { P[i]=i;

printf("P[%d] ",P[i]); for (j=0;j<rz;j++) {

scanf("%d",&allocated[i][j]); avail[j]+=allocated[i][j];

}

}

printf("\nEnter the MAX matrix : \n "); for (i=0;i<rz;i++) {

printf(" %c",(i+97));

avail[i]=ins[i]-avail[i];

}

printf("\n");

for (i=0;i <pno;i++) {

printf("P[%d] ",i); for (j=0;j<rz;j++)

scanf("%d", &MAX[i][j]);

}

printf("\n"); A: d=-1;

for (i=0;i <pno;i++) {

count=0; t=P[i];

for (j=0;j<rz;j++) {

need[t][j] = MAX[t][j]-allocated[t][j]; if(need[t][j]<=avail[j])

count++;

}

if(count==rz) {

output[k++]=P[i]; for (j=0;j<rz;j++)

avail[j]+=allocated[t][j];

} else P[++d]=P[i];

}

if(d!=-1) {

pno=d+1; goto A;

}

printf("\t <"); for (i=0;i<k;i++)

printf(" P[%d] ",output[i]); printf(">");

getch();

}

**OUTPUT:**

