

**Term work**

**on**

**Operating Systems**

**(PCS 506)**

**2021-22**

**Submitted to: Submitted by:**

Ms. Manika Manwal Neeraj Rawat

Assistant Professor University Roll.No:2018514

GEHU, D.Dun Roll. No. /Section: 33/F

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

# PROGRAM – 01

**. Write a C program to demonstrate the use of fork() System call**

## SOURCE CODE

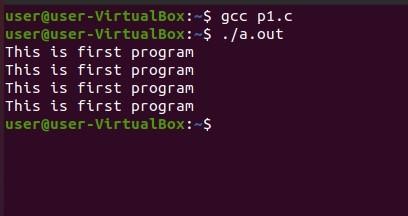
#include <stdio.h> #include <unistd.h>

int main()

{ fork(); fork();

printf("This is first program\n"); return 0; }

## OUTPUT



# PROGRAM – 02

**Objective: Write a C Program to Print sum of even and odd numbers using fork function.**

## SOURCE CODE

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main(){ int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; int

sumOdd = 0, sumEven = 0, n, i;

n = fork();

// Checking if n is not 0 if

(n > 0) { for (i = 0; i < 10; i++) { if (a[i] % 2 == 0) sumEven = sumEven + a[i];

}

printf("Parent process \n");

printf("Sum of even no. is %d\n ",sumEven);

}

// If n is 0 i.e. we are in child process else { for (i = 0; i < 10; i++) {

if (a[i] % 2 != 0) sumOdd = sumOdd + a[i];

}

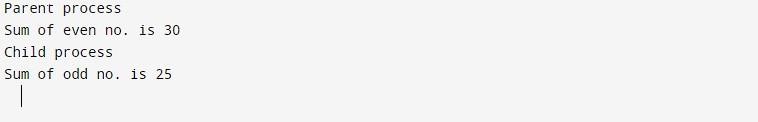
printf("Child process \n");

printf("\nSum of odd no. is %d\n ",sumOdd);

} return

0; }

OUTPUT



**PROGRAM 03**

**Objective: C program to Implement the Orphan Process andZombiProcess.**

### SOURCE CODE

#include<stdio

.h>

#include<unist d.h> int main(){ pid\_t pid; pid=fork(); if(pid==0){ sleep(6); printf("\n I m Child. My PID = %d And PPID = %d",

getpid(),getppid());

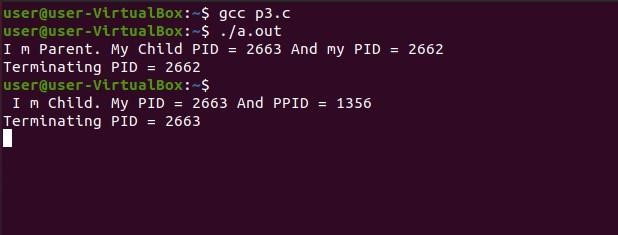
} else{ printf("I m Parent. My Child PID = %d And my PID = %d",pid,getpid());

}

printf("\nTerminating PID = %d\n",getpid());

return 0;

### OUTPUT



### SOURCE CODE

#include<stdio.h>

#include<unistd.h> #include<stdlib.h> int main(){ pid\_t pid; pid=fork(); if(pid!=0){ printf("Parent is Sleeping for 30 sec \n\n"); sleep(50); printf("Both the Process

Exits\n"); } el

se

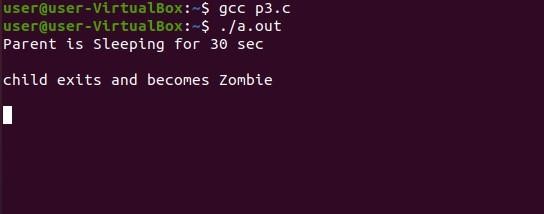
{

printf("child exits and becomes Zombie \n\n"); exit(0);

}

}

## OUTPUT



# PROGRAM – 04

Objective: Write a C Program to implement First Come First Serve CPU Scheduling.

## SOURCE CODE

#include<stdio.h> int main() {

int AT[10],BT[10],WT[10],TT[10],n;

int burst=0,cmpl\_T; float

Avg\_WT,Avg\_TT,Total=0;

printf("Enter number of the process\n");

scanf("%d",&n); printf("Enter Arrival time and Burst time of the process\n");

printf("AT\tBT\n"); for(int i=0;i<n;i++)

{

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

}

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

{ if(i==0)

WT[i]=AT[i]; else WT[i]=burst-AT[i]; burst+=BT[i]; Total+=WT[i];

}

Avg\_WT=Total/n; cmpl\_T=0; Total=0;

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

{

cmpl\_T+=BT[i]; TT[i]=cmpl\_T-AT[i];

Total+=TT[i];

}

Avg\_TT=Total/n; printf("Process ,Waiting\_time ,TurnA\_time\n"); for(int i=0;i<n;i++)

{

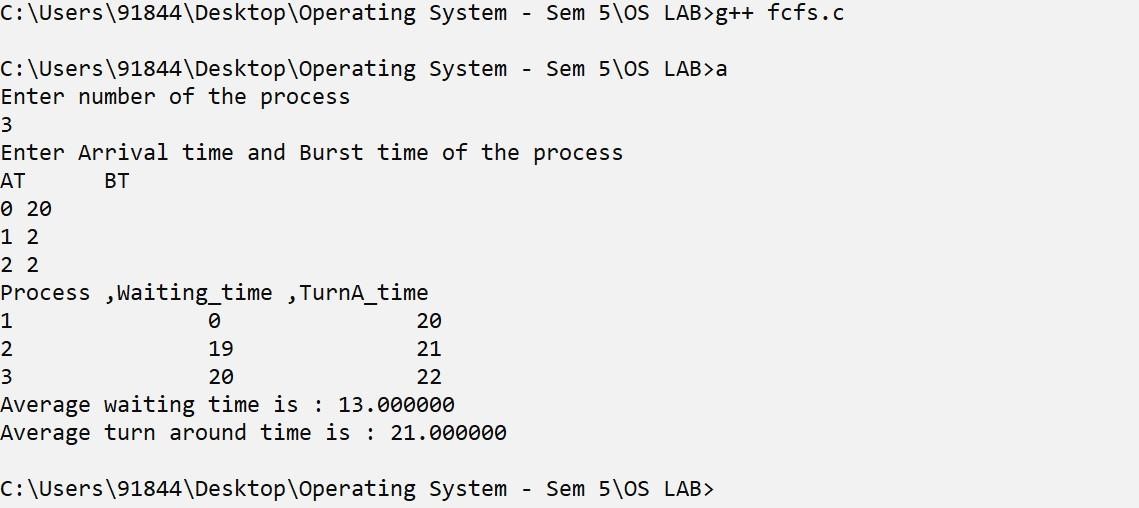
printf("%d\t\t%d\t\t%d\n",i+1,WT[i],TT[i]);

}

printf("Average waiting time is : %f\n",Avg\_WT);

printf("Average turn around time is : %f\n",Avg\_TT); return 0; }

## OUTPUT



# PROGRAM- 05

Objective: Write a C Program to implement Round Robin CPU Scheduling.

## SOURCE CODE

#include<stdio.h> int main() { 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(" Total number of process in the system: ");

scanf("%d", &NOP); y

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

{

printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);

printf(" Arrival time is: \t"); scanf("%d", &at[i]); printf(" \nBurst time is: \t"); scanf("%d", &bt[i]); temp[i] = bt[i];

}

printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);

printf("\n Process No \t\t Burst Time \t\tTAT \t\t Waiting Time "); for(sum=0,

i = 0; y!=0; )

{if(temp[i] <= quant && temp[i] > 0)

{

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--; printf("\nProcess No[%d] \t\t %d\t\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;

} }

avg\_wt = wt \* 1.0/NOP; avg\_tat

= tat \* 1.0/NOP; printf("\n

Average Turn Around Time:

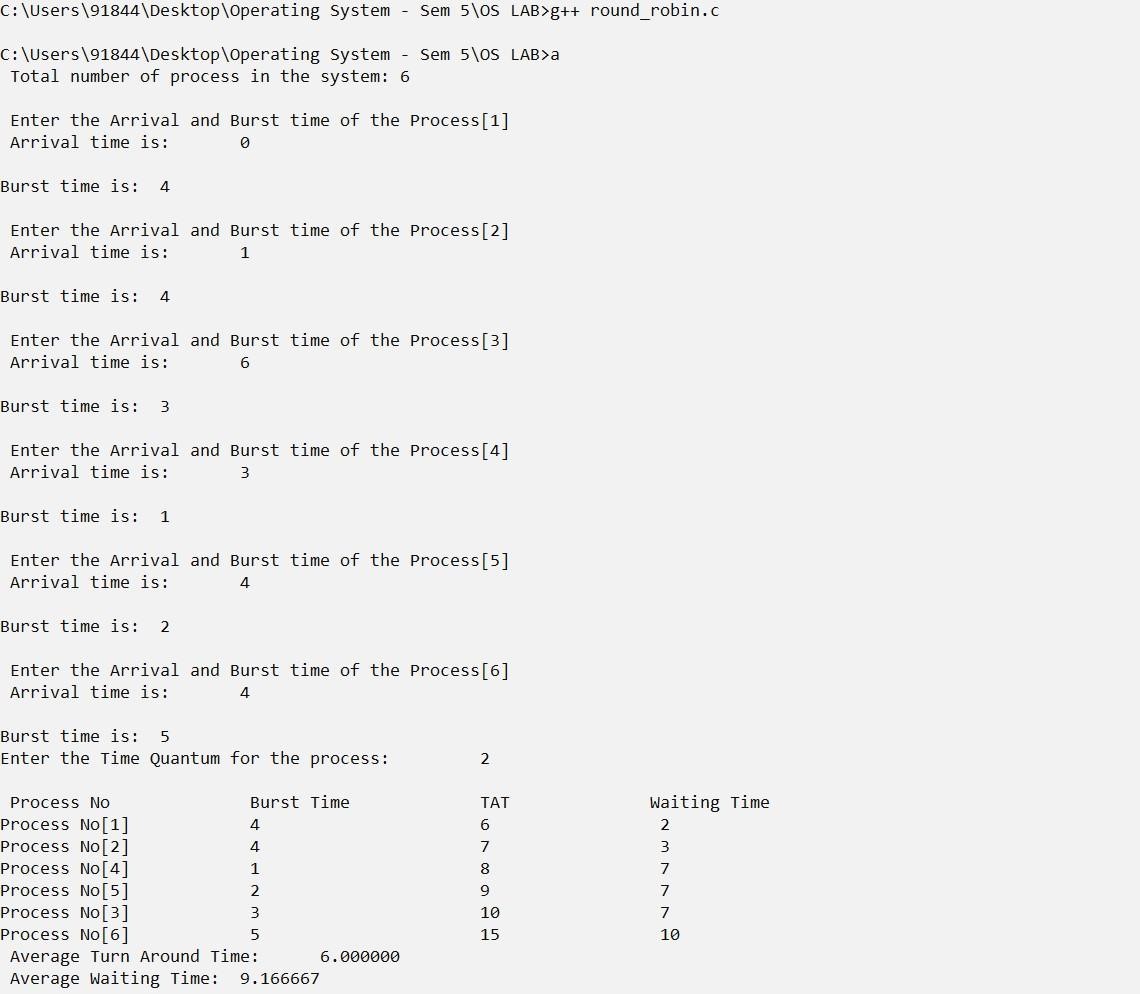
\t%f", avg\_wt); printf("\n Average Waiting Time: \t%f",

avg\_tat);

return 0;



**OUTPUT**



# PROGRAM 06

Objective: Write a C Program to implement Shortest Job First CPU Scheduling (nonPreemptive)

## SOURCE CODE

#include<stdio.h> int main(){

int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0; int bt[10],temp,j,at[10],wt[10],tt[10],ta=0,sum=0; float wavg=0,tavg=0,tsum=0,wsum=0;

printf(" -------Shortest Job First Scheduling ( NP )-------\n"); printf("\nEnter

the No. of processes : ");

scanf("%d",&n); for(i=0;i<n;i++){ printf("\tEnter the burst time of %d process : ",i+1); scanf(" %d",&bt[i]);

printf("\tEnter the arrival time of %d process : ",i+1);

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

}

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

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

temp=p[j]; p[j]=p[i]; p[i]=temp; temp=at[j]; at[j]=at[i]; at[i]=temp; temp=bt[j]; bt[j]=bt[i]; bt[i]=temp;

}

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

{

btime=btime+bt[j];

min=bt[k]; for(i=k;i<n;i++){

if (btime>=at[i] && bt[i]<min){

temp=p[k]; p[k]=p[i]; p[i]=temp; temp=at[k]; at[k]=at[i]; at[i]=temp; temp=bt[k]; bt[k]=bt[i]; bt[i]=temp;

}

} k++; } wt[0]=0; for(i=1;i

<n;i++)

{

sum=su

m+bt[i-

1]; wt[i]=sum-at[i]; wsum=wsum+wt[i];

}

wavg=(wsum/n); for(i=0;i<n;i++)

{ ta=ta+bt[i];

tt[i]=ta-at[i]; tsum=tsum+tt[i];

} tavg=(tsum/n);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n RESULT:-");

printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn around" ); for(i=0;i<n;i++)

{

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

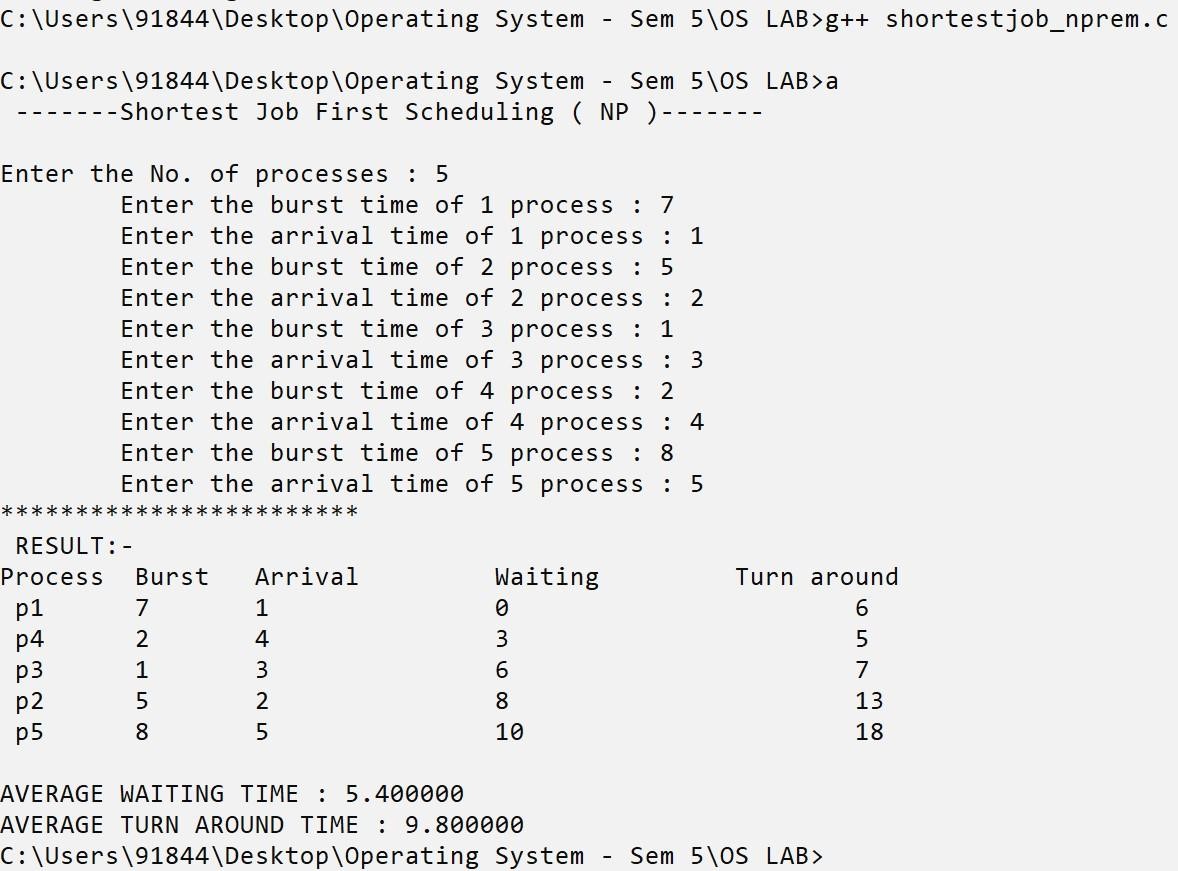
}

printf("\n\nAVERAGE WAITING TIME : %f",wavg);

printf("\nAVERAGE TURN AROUND TIME : %f",tavg); return 0; }



**OUTPUT**



# PROGRAM 07

Objective: Write a C Program to implement Shortest Job First CPU Scheduling (preemptive).

## SOURCE CODE

#include<stdio.h> struct process

{

int WT,AT,BT,TAT;

}; struct process a[10]; int main() {

int n,temp[10]; int

count=0,t=0,short\_P;

float total\_WT=0, total\_TAT=0,Avg\_WT,Avg\_TAT; printf("Enter the number of the process\n");

scanf("%d",&n); printf("Enter the arrival time and burst time of the process\n");

printf("AT\t BT\n"); for(int i=0;i<n;i++)

{

scanf("%d%d",&a[i].AT,&a[i].BT);

temp[i]=a[i].BT; } a[9].BT=10000; for(t=0;count!=n;t++) { short\_P=9; for(int

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

{

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

{ short\_P=i;

}

}

a[short\_P].BT=a[short\_P].BT-1; if(a[short\_P].BT==0)

{ 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\_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;

printf("\nRESULT........\n"); printf("Id WT TAT\n");

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

{

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

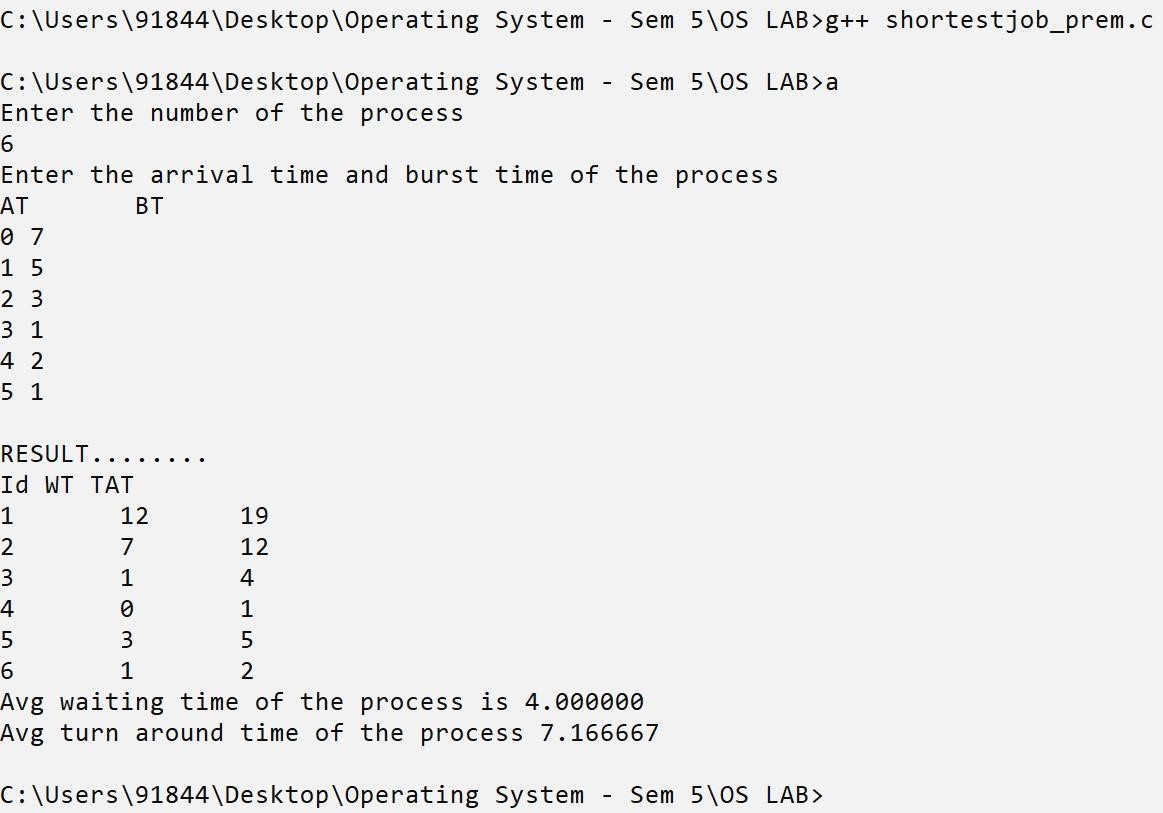
}

printf("Avg waiting time of the process is %f\n",Avg\_WT);

printf("Avg turn around time of the process %f\n",Avg\_TAT); }



**OUTPUT**



# PROGRAM 08

Objective: Write a C Program to implement priority-based scheduling Algorithm.

## SOURCE CODE

#include<stdio.h> int main() { int

bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,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=%d",avg\_wt); printf("\nAverage Turnaround Time=%d\n",avg\_tat); return 0;

}



**OUTPUT**

