EX NO:

DATE:

IMPLEMENTATION OF PRIORITY SCHEDULING

AIM

To schedule snapshot of processes queued according to Priority scheduling.

ALGORITHM

```
Step 1: Define an array of structure process with members pid, btime, pri, wtime & ttime.
Step 2:  Get length of the ready queue, i.e., number of process (say n)
Step 3: Obtain btime and pri for each process.
Step 4:  Sort the processes according to their pri in ascending order.
    4.1: If two process have same pri, then FCFS is used to resolve the tie.
Step 5: The wtime for first process is 0.
Step 6: Compute wtime and ttime for each process as:
    6.1: wtimei+1 = wtimei + btimei
    6.2: ttimei = wtimei + btimei
Step 7: Compute average waiting time awat and average turn around time atur
Step 8: Display the btime, pri, ttime and wtime for each process.
Step 9: Display GANTT chart for the above scheduling
Step 10: Display awat and atur
Step 11: Stop
```

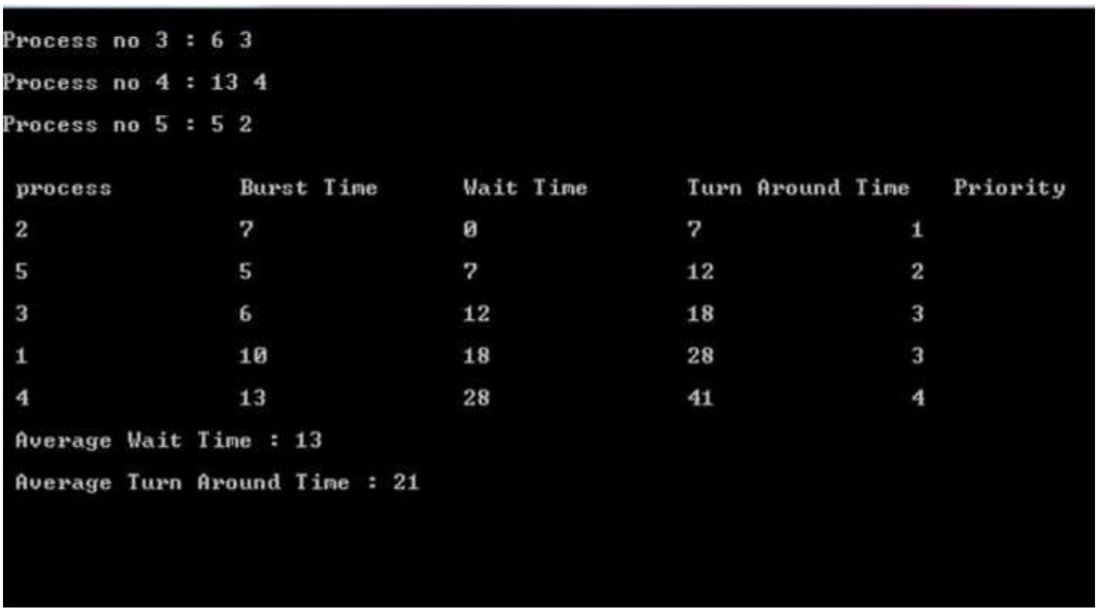
PROGRAM

```
#include<stdio.h>
#include<conio.h>
int main()
{
int x,n,p[10],pp[10],bt[10],w[10],t[10],awt,atat,i,j;
printf("Enter the number of process : ");
```

```
scanf("%d",&n);
printf("\n Enter burst time and priority: \n");
for(i=0;i<n;i++)
{
printf("\nProcess no %d : ",i+1);
scanf("%d %d",&bt[i],&pp[i]);
p[i]=i+1;
}
for(i=0;i<n-1;i++)
{
for(j=i+1;j<n;j++)
{
if(pp[i]>pp[j])
{
x=pp[i];
pp[i]=pp[j];
pp[j]=x;
x=bt[i];
bt[i]=bt[j];
bt[j]=x;
x=p[i];
p[i]=p[j];
p[j]=x;
}
}
}
w[0]=0;
awt=0;
t[0]=bt[0];
atat=t[0];
```

```
for(i=1;i<n;i++)
{
w[i]=t[i-1];
awt+=w[i];
t[i]=w[i]+bt[i];
atat+=t[i];
}
printf("\n\n process \t Burst Time \t Wait Time \t Turn Around Time Priority \n");
for(i=0;i<n;i++)
printf("\n %d \t\t %d \t\t %d \t\t %d \t\t %d \n",p[i],bt[i],w[i],t[i],pp[i]);
awt/=n;
atat/=n;
printf("\n Average Wait Time : %d \n",awt);
printf("\n Average Turn Around Time : %d \n",atat);
getch();
return 0;
}
```

OUTPUT



```
Process no 3 : 6 3
Process no 4 : 13 4
Process no 5 : 5 2

 process         Burst Time      Wait Time       Turn Around Time   Priority
 2               7               0               7               1
 5               5               7               12              2
 3               6               12              18              3
 1               10              18              28              3
 4               13              28              41              4
 Average Wait Time : 13
 Average Turn Around Time : 21
```

EX NO:

DATE:

IMPLEMENTATION OF ROUND ROBIN SCHEDULING

AIM

To schedule snapshot of processes queued according to Round robin scheduling**.**

ALGORITHM

Step 1: Get length of the ready queue, i.e., number of process (say n)

Step 2: Obtain Burst time Bi for each processes Pi.

Step 3: Get the time slice per round, say TS

Step 4: Determine the number of rounds for each process.

Step 5: The wait time for first process is 0.

Step 6: If Bi &gt; TS then process takes more than one round. Therefore turnaround and waiting time should include the time spent for other remaining processes in the same round.

Step 7: Calculate average waiting time and turn around time

Step 8: Display the GANTT chart that includes

8.1: order in which the processes were processed in progression of rounds

8.2: Turnaround time Ti for each process in progression of rounds.

Step 9: Display the burst time, turnaround time and wait time for each process (in order of rounds they were processed).

Step 10: Display average wait time and turnaround time

Step 11: Stop

PROGRAM

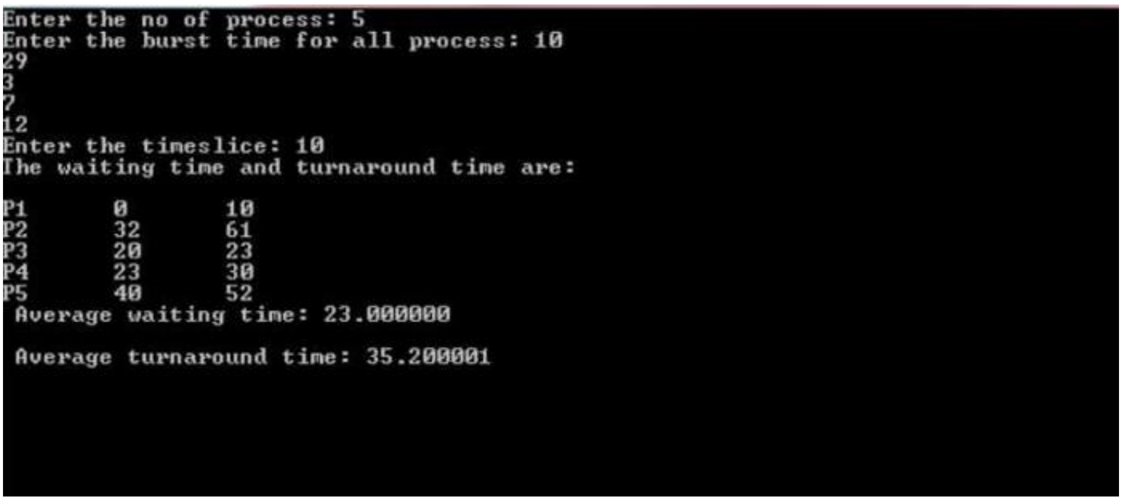
```
#include <stdio.h>

#include<conio.h>

int bt[10],wt[10],tat[10];

int timeslice,n;
```

```
void wt_and_tat(){
int i,j;
int temp=0;
int flag=1;
int c;
int bt1[10];
for(i=0;i<n;i++)
bt1[i]=bt[i];
for(i=0;i<n;i++){
wt[i]=0;
tat[i]=0;
}
while(flag){
for(i=0;i<n;i++){
if(bt[i]==0)
continue;
if(bt[i]>=timeslice){
bt[i]-=timeslice;
temp+=timeslice;
tat[i]=temp;
}
else if(bt[i]<timeslice){
temp+=bt[i];
tat[i]=temp;
bt[i]=0;
}
}
c=0;
for(j=0;j<n;j++)
if(bt[j]!=0){
```

```
c=1;
break;
}
if(c==0)
flag=0;
}
for(i=0;i<n;i++)
wt[i]=tat[i]-bt1[i];
}
void awt_and_atat(){
float awt,atat;
int twt=0,ttat=0;
int i,j;
printf("The waiting time and turnaround time are:\n ");
for(i=0;i<n;i++){
printf("\nP%d\t%d\t%d",i+1,wt[i],tat[i]);
twt+=wt[i];
ttat+=tat[i];
}
awt=(float)twt/n;
atat=(float)ttat/n;
printf("\n Average waiting time: %f\n",awt);
printf("\n Average turnaround time: %f",atat);
}
int main()
{
int i;
printf("Enter the no of process: ");
scanf("%d",&n);
printf("Enter the burst time for all process: ");
```

```
for(i=0;i<n;i++)
scanf("%d",&bt[i]);
printf("Enter the timeslice: ");
scanf("%d",&timeslice);
wt_and_tat();
awt_and_atat();
getch();
return 0;
}
```

OUTPUT



```
Enter the no of process: 5
Enter the burst time for all process: 10
29
3
7
12
Enter the timeslice: 10
The waiting time and turnaround time are:

P1      0       10
P2      32      61
P3      20      23
P4      23      30
P5      40      52
 Average waiting time: 23.000000

 Average turnaround time: 35.200001
```

OBSERVATION (20)	
RECORD (5)	
TOTAL (25)	

RESULT

Thus waiting time &amp; turnaround time for processes based on Priority scheduling and Round Robin Scheduling was computed and the average waiting time was determined.