CLASS ASSIGNMENT

Course code: CSE316

Course Title: Operating System

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NAME : SAIKIRAN Y

Reg No: 11716260

Section: EE033

Roll No : B68

Question Assigned: 5

Submitted to : NAHIDA NAZIR

PROGRAM DECRIPTION:

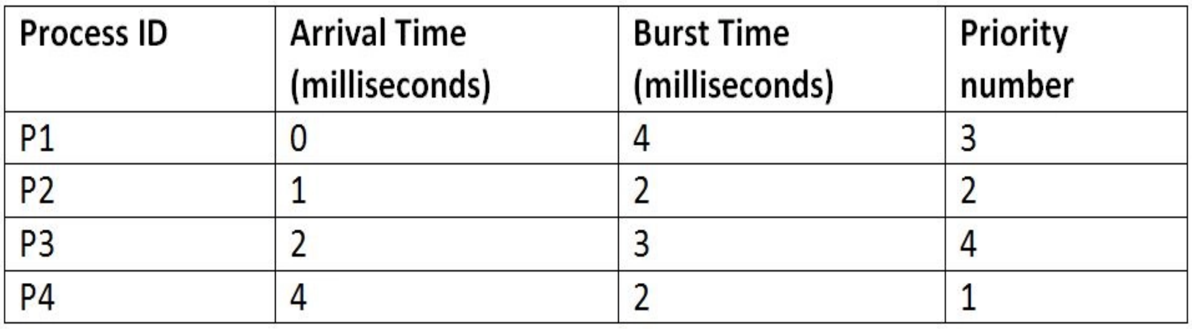
Consider a scheduling approach which is non pre- emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times. The priority can be computed as: Priority = 1+ Waiting time / Estimated run time Write a program to implement such an algorithm.

EXPLANATION:

By taking the below example and problem solving this algorithm:

The scheduling depends upon the priority of the processes rather than its burst time. So, the processes, in this case, must also have the priority number in its details on the basis of which the OS will schedule it.

- There is only P1 available at time 0, so it will be executed first irrespective of the priority, and it cannot be preempted in between before its completion.
- When it is completed at 4^{th}-time unit, we have all P2, P3, and P4 available. So, they are executed according to their priorities.

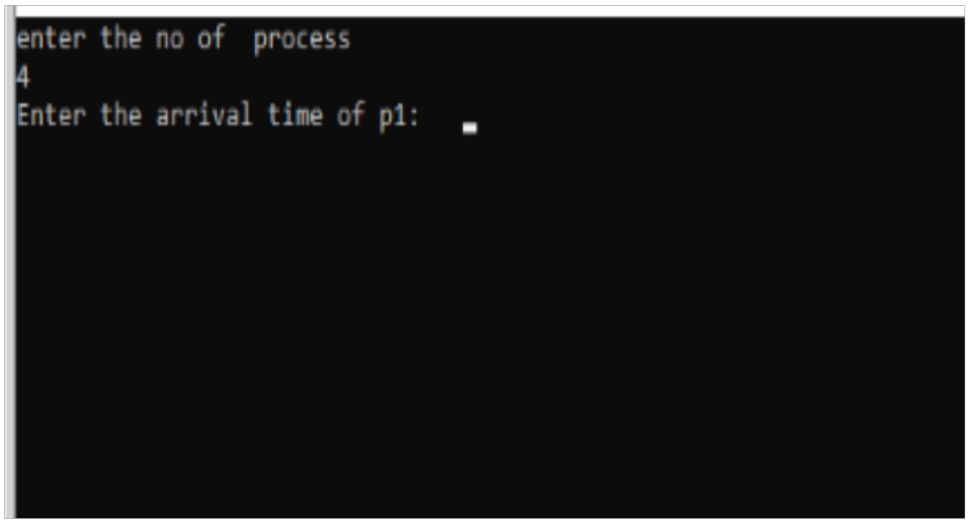


Process ID	Arrival Time (milliseconds)	Burst Time (milliseconds)	Priority number
P1	0	4	3
P2	1	2	2
P3	2	3	4
P4	4	2	1

STEPS FOR EXECUTION:

STEP 1:

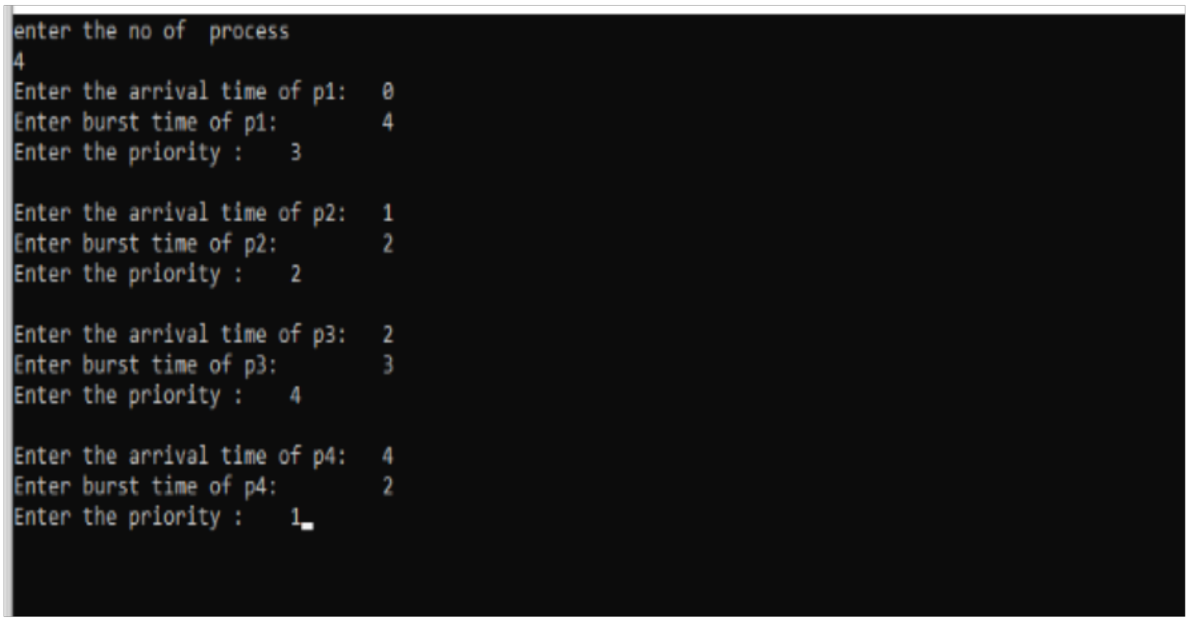
Enter the number of processes:



```
enter the no of  process
4
Enter the arrival time of p1:
```

STEP 2:

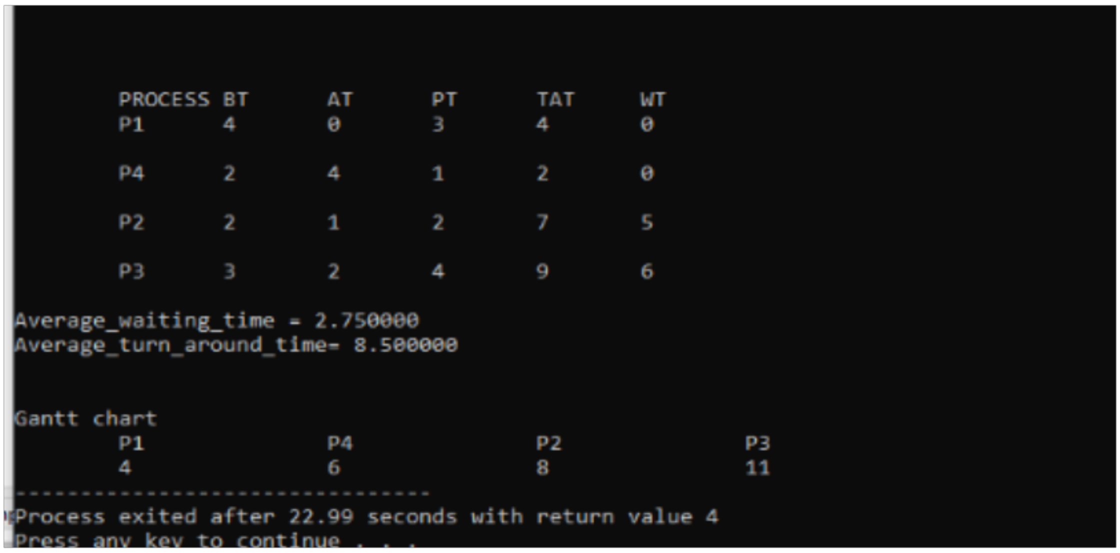
Enter the arrival time, burst time and priority of all the processes



```
enter the no of  process
4
Enter the arrival time of p1:   0
Enter burst time of p1:         4
Enter the priority :    3

Enter the arrival time of p2:   1
Enter burst time of p2:         2
Enter the priority :    2

Enter the arrival time of p3:   2
Enter burst time of p3:         3
Enter the priority :    4

Enter the arrival time of p4:   4
Enter burst time of p4:         2
Enter the priority :    1
```

STEP 3:

By compiling the following code the average waiting time and average turn around time will be calculated.



```
        PROCESS BT      AT      PT      TAT     WT
        P1      4       0       3       4       0

        P4      2       4       1       2       0

        P2      2       1       2       7       5

        P3      3       2       4       9       6
Average_waiting_time = 2.750000
Average_turn_around_time= 8.500000


Gantt chart
        P1              P4              P2              P3
        4               6               8               11
--------------------------------
Process exited after 22.99 seconds with return value 4
Press any key to continue . . .
```

CALCULATION:

Average Waiting Time = Total Waiting Time / Total No. of Processes

= 11 / 4

= 2.75 milliseconds

Average Turn Around Time= Total Turn Around Time / Total No. of Processes

= 34/ 4

= 8.5 milliseconds

CODE:

```
#include<stdio.h>
void main()
{
 int n,i,j,temp,TEMP1,TEMP2,TEMP3,TEMP4;
 float WTSUM=0,TATSUM=0;
 int bt[10],at[10],P[10],ct[10],tat[10],wt[10],pt[10];
 printf("enter the no of  process\n");
 scanf("%d",&n);
 for(i=1;i<=n;i++)
 {
  printf("Enter the arrival time of p%d:\t",i);
  scanf("%d",&at[i]);
  printf("Enter burst time of p%d: \t",i);
  scanf("%d",&bt[i]);
  printf("Enter the priority :\t");
  scanf("%d",&pt[i]);
  P[i]=i;
  printf("\n");
```

```
}
ct[0]=0;
   if(bt[1]>=at[n])
   {
       ct[1]=bt[1]+at[1];
       tat[1]=ct[1]-at[1];
       wt[1]=tat[1]-bt[1];
 WTSUM=wt[1];
 TATSUM=tat[1];
       for(i=2;i<=n;i++)
       {
          for(j=i+1;j<=n;j++)
          {
             if(pt[j]<pt[i])
             {
                TEMP4=pt[i];
                pt[i]=pt[j];
                pt[j]=TEMP4;
                TEMP1=bt[i];
                bt[i]=bt[j];
                bt[j]=TEMP1;
                TEMP2=at[i];
                at[i]=at[j];
                at[j]=TEMP2;
                TEMP3=P[i];
                P[i]=P[j];
                P[j]=TEMP3;
             }
          }
             if(ct[i-1]<at[i])
             {
                temp=at[i]-ct[i-1];
                ct[i]=ct[i-1]+bt[i]+temp;
                TEMP1=bt[i];
             }
             else
          {
            ct[i]=ct[i-1]+bt[i];
          }
          tat[i]=ct[i]-at[i];
          wt[i]=tat[i]-bt[i];
          WTSUM=WTSUM+wt[i]+wt[1];
          TATSUM=TATSUM+tat[i]+tat[1];
    }
  }
        if(at[n]==0)
        {
        ct[0]=0;
        for(i=1;i<=n;i++)
          {
            for(j=i+1;j<=n;j++)
            {
              if(pt[j]<pt[i])
              {
                TEMP4=pt[i];
                pt[i]=pt[j];
                pt[j]=TEMP4;
               TEMP1=bt[i];
               bt[i]=bt[j];
               bt[j]=TEMP1;
              TEMP2=at[i];
              at[i]=at[j];
              at[j]=TEMP2;
              }
            }
          if(ct[i-1]<at[i])
          {

            temp=at[i]-ct[i-1];
            ct[i]=ct[i-1]+bt[i]+temp;
            TEMP1=bt[i];
          }
```

```
               else
               {
                   ct[i]=ct[i-1]+bt[i];
               }
           tat[i]=ct[i]-at[i];
           wt[i]=tat[i]-bt[i];
           WTSUM=WTSUM+wt[i];
          TATSUM=TATSUM+tat[i];
       }
   }
 printf("\n\n\n");
 printf("\tPROCESS\tBT\tAT\tPT\tTAT\tWT\n");
 for(i=1;i<=n;i++)
 {
 printf("\tP%d\t%d\t%d\t%d  \t%d\t%d\n\n",P[i],bt[i],at[i],pt[i],tat[i],wt[i]);
 }
 printf("Average_waiting_time = %f\n",WTSUM/n);
 printf("Average_turn_around_time= %f\n",TATSUM/n);
 printf("\n\n");
 printf("Gantt chart\n");
 for(i=1;i<=n;i++)
 {
  printf("\tP%d\t",P[i]);

 }
 printf("\n");
 for(i=1;i<=n;i++)
 {
 printf("\t%d\t",ct[i]);
 }
}
```

THANK YOU