PROBLEM DEFINITION

Given a set of processes with their arrival and burst time, schedule them for processing in accordance to the highest response ratio amongst all the process in the ready queue and thereby calculate the average turn around time and waiting time.

THEORY

Highest Response Ratio Next (HRRN) is one of the most optimal scheduling algorithms. This is a non-pre-emptive algorithms, i.e., a process cannot be forcibly withdrawn until it completes its execution.

Here, the scheduling is done on the basis of an extra parameter called "Response Ratio". A Response Ratio is calculated for each of the available process and the process with highest Response Ratio is given priority over the others.

Response Ratio is calculated as:

RR=(WT+BT)/BT

Where,

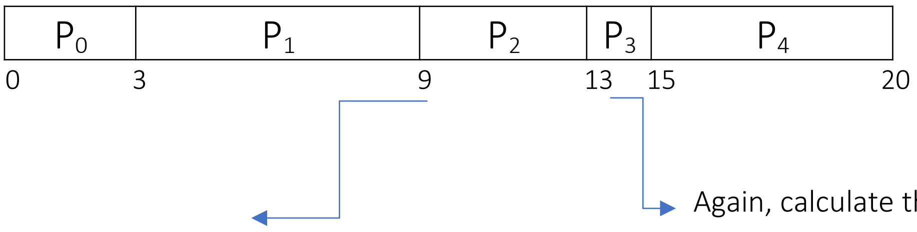
WT=Waiting Time

BT=Burst Time

ILLUSTRATION

Process ID	Arrival Time (AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time TAT=CT-AT	Waiting Time WT=TAT-BT
0	0	3	3	3	0
1	2	6	9	7	1
2	4	4	13	9	5
3	6	5	20	14	9
4	8	2	15	7	5
	Average TAT and WT			8	4

Gantt Chart:



By now, all processes have arrived. Now, choosing next process depends on the RR values of the remaining process.

∴ $RR_2 = ((9-4)+4)/4 = \underline{2.25}$

$RR_3 = ((9-6)+4)/4 = 1.75$

$RR_4 = ((9-8)+2)/2 = 1.5$

–

Again, calculate the RR for remaining processes and choose the process with highest RR.

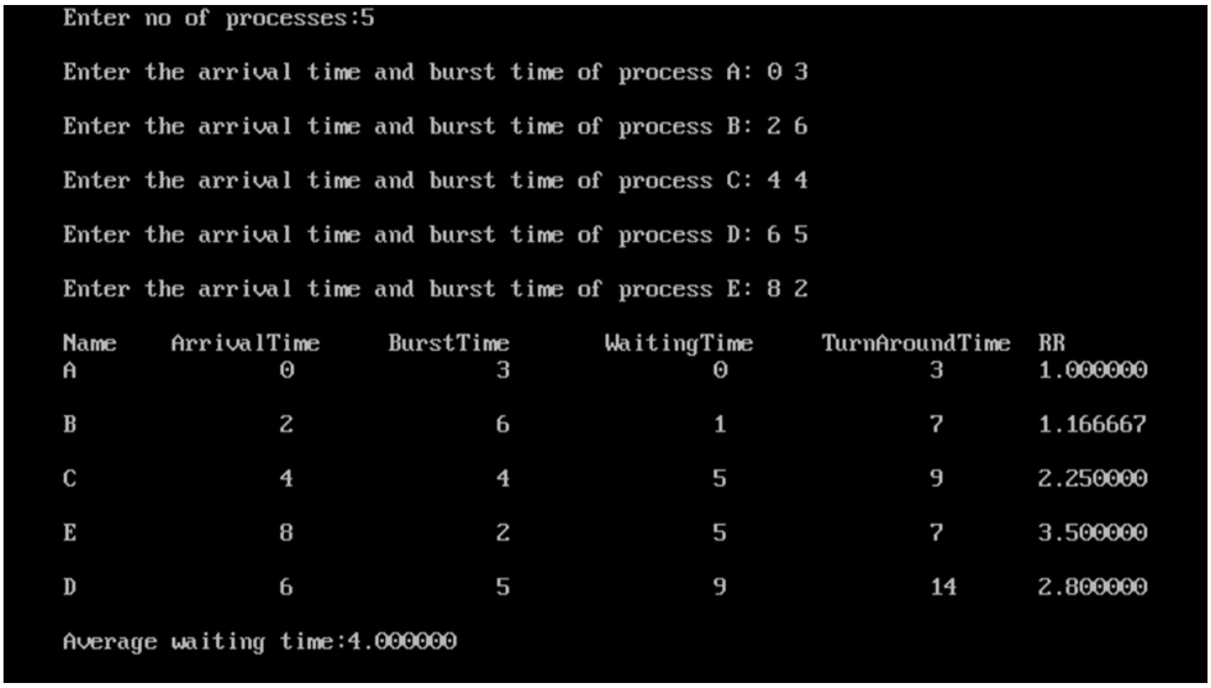
∴ $RR_3 = ((13-6)+5)/5 = 2.4$

$RR_4 = ((13-8)+2)/2 = \underline{3.5}$

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
struct process
{
 char name;
 int at,bt,ct,wt,tt;
 int completed;
 float ntt;
}p[10];
int n;
void sortByArrival()
{
 struct process temp;
 int i,j;
 for(i=0;i<n-1;i++)
 {
  for(j=i+1;j<n;j++)
  {
   if(p[i].at>p[j].at)
   {
    temp=p[i];
    p[i]=p[j];
    p[j]=temp;
   }
  }
 }
}
void main()
{
 int i,j,time,sum_bt=0;
 char c;
 float avgwt=0;
 clrscr();
 printf("Enter no of processes:");
 scanf("%d",&n);
 for(i=0,c='A';i<n;i++,c++)
 {
  p[i].name=c;
  printf("\nEnter the arrival time and burst time of process %c: ",p[i].name);
  scanf("%d%d",&p[i].at,&p[i].bt);
  p[i].completed=0;
  sum_bt+=p[i].bt;
 }
 sortByArrival();
 printf("\nName\tArrivalTime\tBurstTime\tWaitingTime\tTurnAroundTime\tRR";
 for(time=p[0].at;time<sum_bt;)
 {
  float hrr=-9999;
  int loc;
  for(i=0;i<n;i++)
  {
   if(p[i].at<=time && p[i].completed!=1)
   {
    float temp=(p[i].bt + (time-p[i].at))/p[i].bt;
    if(hrr < temp)
    {
     hrr=temp;
     loc=i;
    }
   }
  }
  time+=p[loc].bt;
  p[loc].wt=time-p[loc].at-p[loc].bt;
  p[loc].tt=time-p[loc].at;
  p[loc].ntt=((float)p[loc].tt/p[loc].bt);
  p[loc].completed=1;
  avgwt+=p[loc].wt;
  printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d\t%f",p[loc].name,p[loc].at,p[loc].bt,p[lo
 .wt,p[loc].tt,p[loc].ntt);
 }
 printf("\nAverage waiting time:%f\n",avgwt/n);
 getch();
}
```

OUTPUT



```
Enter no of processes:5

Enter the arrival time and burst time of process A: 0 3

Enter the arrival time and burst time of process B: 2 6

Enter the arrival time and burst time of process C: 4 4

Enter the arrival time and burst time of process D: 6 5

Enter the arrival time and burst time of process E: 8 2

Name    ArrivalTime     BurstTime       WaitingTime     TurnAroundTime  RR
A               0               3               0               3       1.000000

B               2               6               1               7       1.166667

C               4               4               5               9       2.250000

E               8               2               5               7       3.500000

D               6               5               9               14      2.800000

Average waiting time:4.000000
```

Here,RR is the Response Ratio.