```
#include<stdio.h>
void preemptive();
void roundrobin();
struct rrbn
{
char name;
int at,bt,wt,tt,rt;
int completed;
}p[10];
int n;
int q[10];  //queue
int front=-1,rear=-1;
void enqueue(int i)
{
if(rear==10)
printf("overflow");
rear++;
q[rear]=i;
if(front==-1)
front=0;
}
int dequeue()
{
if(front==-1)
printf("underflow");
int temp=q[front];
if(front==rear)
front=rear=-1;
else
front++;
return temp;
}
int isInQueue(int i)
{int k;
for(k=front;k<=rear;k++)
{
if(q[k]==i)
return 1;
}
return 0;
}void sortByArrival()
{
struct rrbn 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;
```

```
}
}
}
int main()
{
 printf("\t*** Fixed priority preemptive Scheduling ***\n");
 preemptive();
 printf("\n \n");
 printf("\t*** Round Robin Scheduling ***\n");
 roundrobin();
}
void preemptive()
{
 int num;
printf("Enter the no. of processes: ");
scanf("%d",&num);
if(num<=0)
 printf("ENTER A MINIMUM OF 1 PROCESS \n");
sortByArrival();
int id[num],bt[num],wt[num],tat[num],p[num],i,j,temp;
for(i=0;i<num;i++)
{
printf("Enter process %d id: ",i+1);
scanf("%d",&id[i]);
printf("Enter process %d burst time: ",i+1);
scanf("%d",&bt[i]);
bt[i]=bt[i]*2;
printf("Enter process %d priority: ",i+1);
scanf("%d",&p[i]);
}
for(i=0;i<num;i++)
{
for(j=i+1;j<num;j++)
{
if(p[i]>p[j])
{
temp=p[i];
p[i]=p[j];
p[j]=temp;
temp=bt[i];
bt[i]=bt[j];
bt[j]=temp;
temp=id[i];
id[i]=id[j];
id[j]=temp;
}
}
wt[i]=0;
}
for(i=0;i<num;i++)
{
for(j=0;j<i;j++)
```

```
{
wt[i]=wt[i]+bt[j];
}
tat[i]=wt[i]+bt[i];
}
float avwt=0,avtat=0;
printf("Process\tP\tBT\tWT\tTAT\n");
for(i=0;i<num;i++)
{
printf("%d\t%d\t%d\t%d\t%d\n",id[i],p[i],bt[i],wt[i],tat[i]);
avwt=avwt+wt[i];
avtat=avtat+tat[i];
}
printf("Average Waiting Time: %f\n",avwt/num);
printf("\nAverage Turnaround Time: %f",avtat/num);
}
void roundrobin()
{
 int i,j,time=0,sum_bt=0,tq;
 char c;
 float avgwt=0;
  printf("Enter no of processes: ");
  scanf("%d",&n);
  for(i=0,c='A';i<n;i++,c++)
  {
   p[i].name=c;
   printf("\nEnter arrival time [process] %c: ",p[i].name);
   scanf("%d",&p[i].at);
   printf("Enter burst time [process] %c: ",p[i].name);
   scanf("%d",&p[i].bt);
  p[i].bt=p[i].bt*2;
   p[i].rt=p[i].bt;
   p[i].completed=0;
   sum_bt+=p[i].bt;
 }
printf("\nEnter time quantum: ");
scanf("%d",&tq);
if(tq<=0)
 printf("ENTER A MINIMUM OF 1 TIME QUANTUM \n");
sortByArrival();
enqueue(0);
printf("Process execution order: ");
for(time=p[0].at;time<sum_bt;)
{
 i=dequeue();
 if(p[i].rt<=tq)
 {
  time+=p[i].rt;
  p[i].rt=0;
  p[i].completed=1;
     printf(" %c ",p[i].name);
        p[i].wt=time-p[i].at-p[i].bt;
```

```
        p[i].tt=time-p[i].at;
        for(j=0;j<n;j++)
        {
         if(p[j].at<=time && p[j].completed!=1&& isInQueue(j)!=1)
            {
            enqueue(j);
            }
        }
    }
    else
    {
    time+=tq;
    p[i].rt-=tq;
    printf(" %c ",p[i].name);
    for(j=0;j<n;j++)
        {
        if(p[j].at<=time && p[j].completed!=1&&i!=j&& isInQueue(j)!=1)
        {
         enqueue(j);
        }
        }
    enqueue(i);
    }
}
printf("\nName\tArrival Time\tBurst Time\tWaiting Time\tTurnAround Time\t");
for(i=0;i<n;i++)
{avgwt+=p[i].wt;
printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d\t\t%f",p[i].name,p[i].at,p[i].bt,p[i].wt,p[i].tt);
}
printf("\nAverage waiting time:%f\n",avgwt/n);
}
```