Priority and Round Robin CPU Scheduling

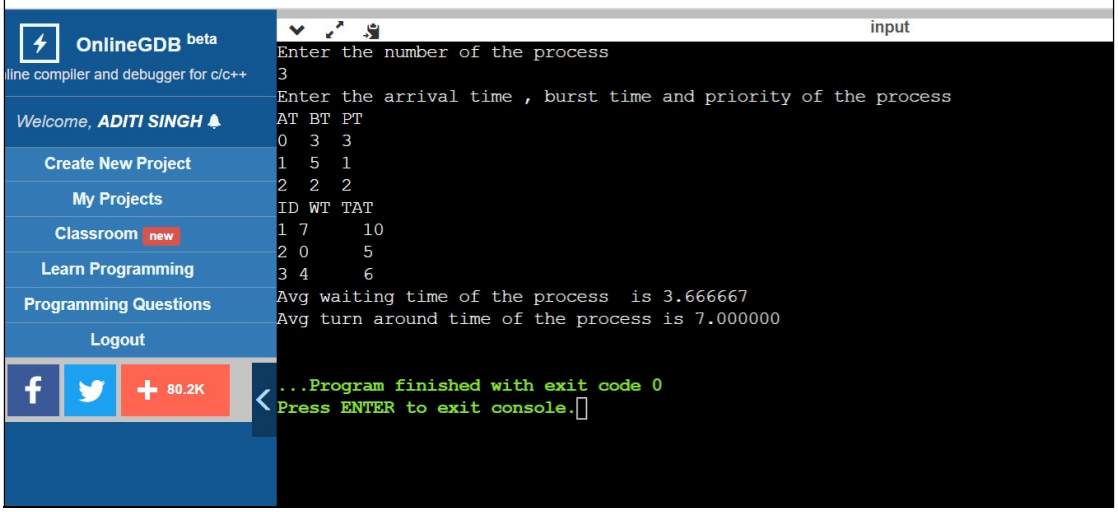
Sl no	Question
1	Write a program in C to implement Priority CPU scheduling (preemptive)(o/p-response time, turnaround time , waiting time, average waiting time.)
2	Write a program in C to implement Round Robin CPU scheduling(o/p-response time, turnaround time , waiting time, average waiting time.)

Solution:-

1.)
```
//Write a program in C to implement Priority CPU scheduling
(preemptive)(o/p-response time, turnaround time , waiting time,
average waiting time.)
#include<stdio.h>
struct process
{
   int WT,AT,BT,TAT,PT;
};

struct process a[10];

int main()
{
   int n,temp[10],t,count=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 , burst time and priority of the
process\n");
   printf("AT BT PT\n");
   for(int i=0;i<n;i++)
   {
      scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

      // copying the burst time in
      // a temp array fot futher use
      temp[i]=a[i].BT;
   }
```

```
// we initialize the burst time
// of a process with maximum
a[9].PT=10000;

for(t=0;count!=n;t++)
{
   short_p=9;
   for(int i=0;i<n;i++)
   {
      if(a[short_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)
      {
         short_p=i;
      }
   }

   a[short_p].BT=a[short_p].BT-1;

   // if any process is completed
   if(a[short_p].BT==0)
   {
      // one process is completed
      // so count increases by 1
      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 calculation
      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;

// printing of the answer
printf("ID WT TAT\n");
for(int i=0;i<n;i++)
{
   printf("%d %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 is %f\n",Avg_TAT);
```

```
    return 0;
}
```

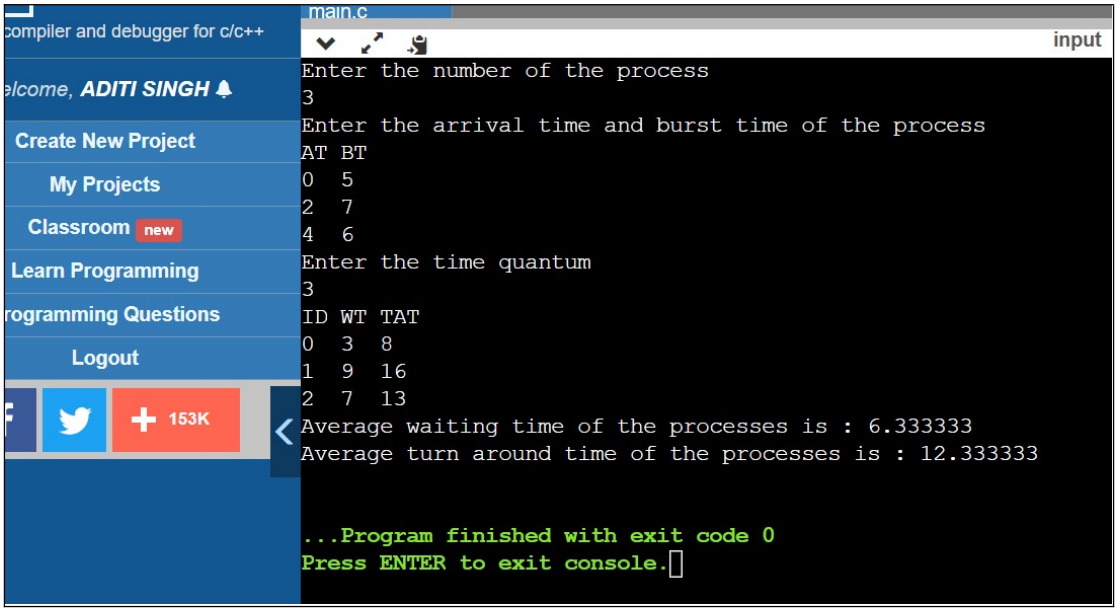


2.) //Write a program in C to implement Round Robin CPU scheduling(o/p-response time, turnaround time , waiting time,average waiting time.)

```
#include<stdio.h>
struct process
{
   int id,AT,BT,WT,TAT;
};

struct process a[10];

// declaration of the ready queue
int queue[100];
int front=-1;
int rear=-1;

// function for insert the element
// into queue
void insert(int n)
{
   if(front==-1)
    front=0;
   rear=rear+1;
   queue[rear]=n;
}

// function for delete the
// element from queue
int delete()
```

```
{
   int n;
   n=queue[front];
   front=front+1;
   return n;
}
int main()
{
   int n,TQ,p,TIME=0;
   int temp[10],exist[10]={0};
   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 BT\n");
   for(int i=0;i<n;i++)
   {
      scanf("%d%d",&a[i].AT,&a[i].BT);
      a[i].id=i;
      temp[i]=a[i].BT;
   }
   printf("Enter the time quantum\n");
   scanf("%d",&TQ);
   // logic for round robin scheduling

   // insert first process
   // into ready queue
   insert(0);
   exist[0]=1;
   // until ready queue is empty
   while(front<=rear)
   {
      p=delete();
      if(a[p].BT>=TQ)
      {
         a[p].BT=a[p].BT-TQ;
         TIME=TIME+TQ;
      }
      else
      {
         TIME=TIME+a[p].BT;
         a[p].BT=0;
      }


      //if process is not exist
```

```
    // in the ready queue even a single
    // time then insert it if it arrive
    // at time 'TIME'
    for(int i=0;i<n;i++)
    {
       if(exist[i]==0 && a[i].AT<=TIME)
       {
          insert(i);
          exist[i]=1;
       }
    }
    // if process is completed
    if(a[p].BT==0)
    {
       a[p].TAT=TIME-a[p].AT;
       a[p].WT=a[p].TAT-temp[p];
       total_tat=total_tat+a[p].TAT;
       total_wt=total_wt+a[p].WT;
    }
    else
    {
       insert(p);
    }
  }

  Avg_TAT=total_tat/n;
  Avg_WT=total_wt/n;

  // printing of the answer
  printf("ID WT TAT\n");
  for(int i=0;i<n;i++)
  {
     printf("%d  %d  %d\n",a[i].id,a[i].WT,a[i].TAT);
  }
  printf("Average waiting time of the processes is : %f\n",Avg_WT);
  printf("Average turn around time of the processes is : %f\n",Avg_TAT);
  return 0;
}
```


compiler and debugger for c/c++
lcome, ADITI SINGH
Create New Project
My Projects
Classroom new
Learn Programming
rogramming Questions
Logout
153K
main.c
input
Enter the number of the process
3
Enter the arrival time and burst time of the process
AT BT
0 5
2 7
4 6
Enter the time quantum
3
ID WT TAT
0 3 8
1 9 16
2 7 13
Average waiting time of the processes is : 6.333333
Average turn around time of the processes is : 12.333333
...Program finished with exit code 0
Press ENTER to exit console.