***MAINAK CHATTOPADHYAY***
***21BAI1217***

# ***OS LAB - L27+L28***

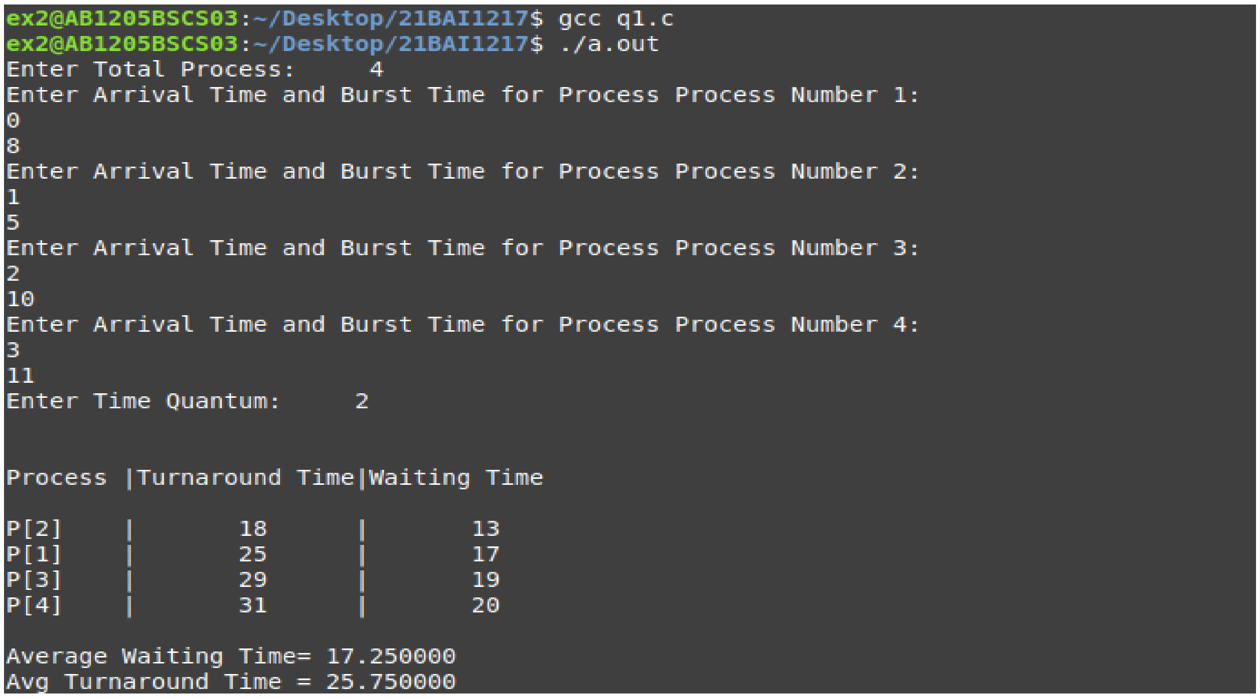
1. Consider a ticket counter in railway station where the person has to stand in a queue for booking the tickets. The person who arrives first in the queue can buy first 2 then only the next person. This will continue until the last person in the queue purchases the ticket. The order of arriving in the queue decides the chance to buy the ticket. Write a program in C to implement the appropriate scheduling algorithm to calculate the average waiting time and average turn around time for the entire booking process. Display the order of person availing in queue and also the waiting time and turn around time for each process.

**CODE-**

```
#include<stdio.h>

int main()
{

 int count,j,n,time,remain,flag=0,time_quantum;
 int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
 printf("Enter Total Process:\t ");
 scanf("%d",&n);
 remain=n;
 for(count=0;count<n;count++)
 {
  printf("Enter Arrival Time and Burst Time for Process Process Number %d: \n",count+1);
  scanf("%d",&at[count]);
  scanf("%d",&bt[count]);
  rt[count]=bt[count];
 }
 printf("Enter Time Quantum:\t");
 scanf("%d",&time_quantum);
 printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
 for(time=0,count=0;remain!=0;)
 {
  if(rt[count]<=time_quantum && rt[count]>0)
  {
   time+=rt[count];
   rt[count]=0;
   flag=1;
  }
  else if(rt[count]>0)
  {
   rt[count]-=time_quantum;
```

```
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
      wait_time+=time-at[count]-bt[count];
      turnaround_time+=time-at[count];
      flag=0;
    }
    if(count==n-1)
      count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f\n",turnaround_time*1.0/n);

  return 0;
}
```

## OUTPUT



```
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ gcc q1.c
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ ./a.out
Enter Total Process:     4
Enter Arrival Time and Burst Time for Process Process Number 1:
0
8
Enter Arrival Time and Burst Time for Process Process Number 2:
1
5
Enter Arrival Time and Burst Time for Process Process Number 3:
2
10
Enter Arrival Time and Burst Time for Process Process Number 4:
3
11
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[2]    |       18      |       13
P[1]    |       25      |       17
P[3]    |       29      |       19
P[4]    |       31      |       20

Average Waiting Time= 17.250000
Avg Turnaround Time = 25.750000
```

2. (i) Consider an online delivery app which delivers the program booked online by the customers. This app always chooses the nearest customer first for delivering the product and then the next. If 2 customers are booking at the same time, the nearest customer will be chosen first. Use an appropriate scheduling algorithm in order to deliver the products in correct

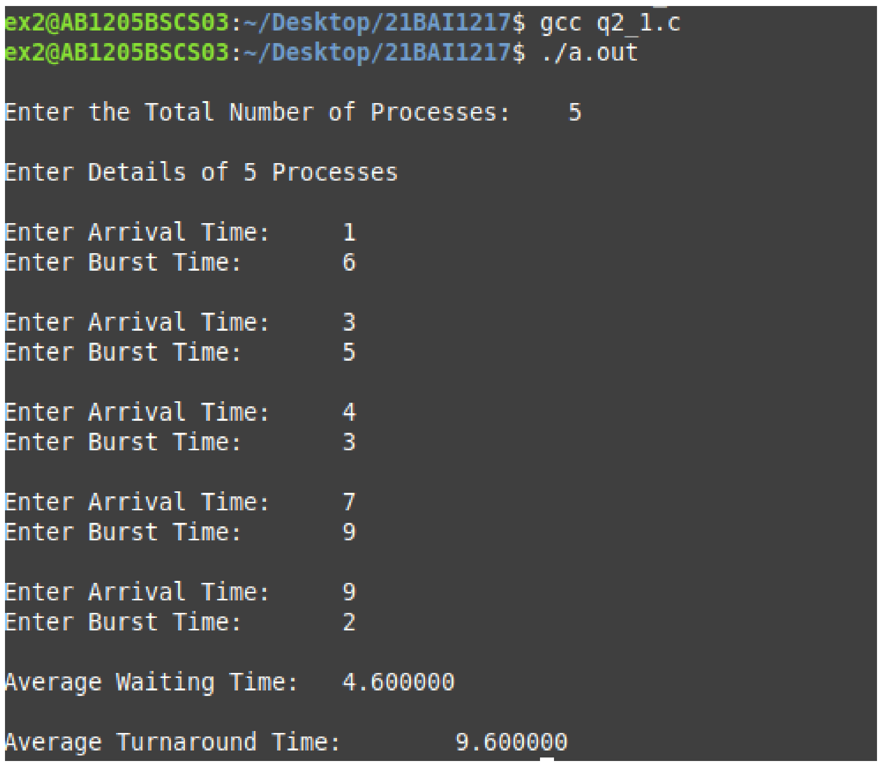
| Customer | Booking Time | Nearest list |
|---|---|---|
| C1 | 1 | 6 |
| C2 | 3 | 5 |
| C3 | 4 | 3 |
| C4 | 7 | 9 |
| C5 | 9 | 2 |

CODE -

```
#include <stdio.h>

int main()
{
    int arrival_time[10], burst_time[10], temp[10];
    int i, smallest, count = 0, time, limit;
    double wait_time = 0, turnaround_time = 0, end;
    float average_waiting_time, average_turnaround_time;
    printf("\nEnter the Total Number of Processes:\t");
    scanf("%d", &limit);
    printf("\nEnter Details of %d Processes\n", limit);
    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Enter Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }
    burst_time[9] = 9999;
    for(time = 0; count != limit; time++)
    {
        smallest = 9;
        for(i = 0; i < limit; i++)
```

```
            {
                if(arrival_time[i] <= time && burst_time[i] < burst_time[smallest] && burst_time[i] > 0)
                {
                    smallest = i;
                }
            }
            burst_time[smallest]--;
            if(burst_time[smallest] == 0)
            {
                count++;
                end = time + 1;
                wait_time = wait_time + end - arrival_time[smallest] - temp[smallest];
                turnaround_time = turnaround_time + end - arrival_time[smallest];
            }
        }
        average_waiting_time = wait_time / limit;
        average_turnaround_time = turnaround_time / limit;
        printf("\nAverage Waiting Time:\t%lf\n", average_waiting_time);
        printf("\nAverage Turnaround Time:\t%lf\n", average_turnaround_time);
        return 0;
    }
```

OUTPUT-



```
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ gcc q2_1.c
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ ./a.out

Enter the Total Number of Processes:    5

Enter Details of 5 Processes

Enter Arrival Time:     1
Enter Burst Time:       6

Enter Arrival Time:     3
Enter Burst Time:       5

Enter Arrival Time:     4
Enter Burst Time:       3

Enter Arrival Time:     7
Enter Burst Time:       9

Enter Arrival Time:     9
Enter Burst Time:       2

Average Waiting Time:   4.600000

Average Turnaround Time:        9.600000
```

(ii) If the customer has a discount coupon in the priority based then the customer will get the chance for fastest delivery.

Discount Priorities:

5

4

1

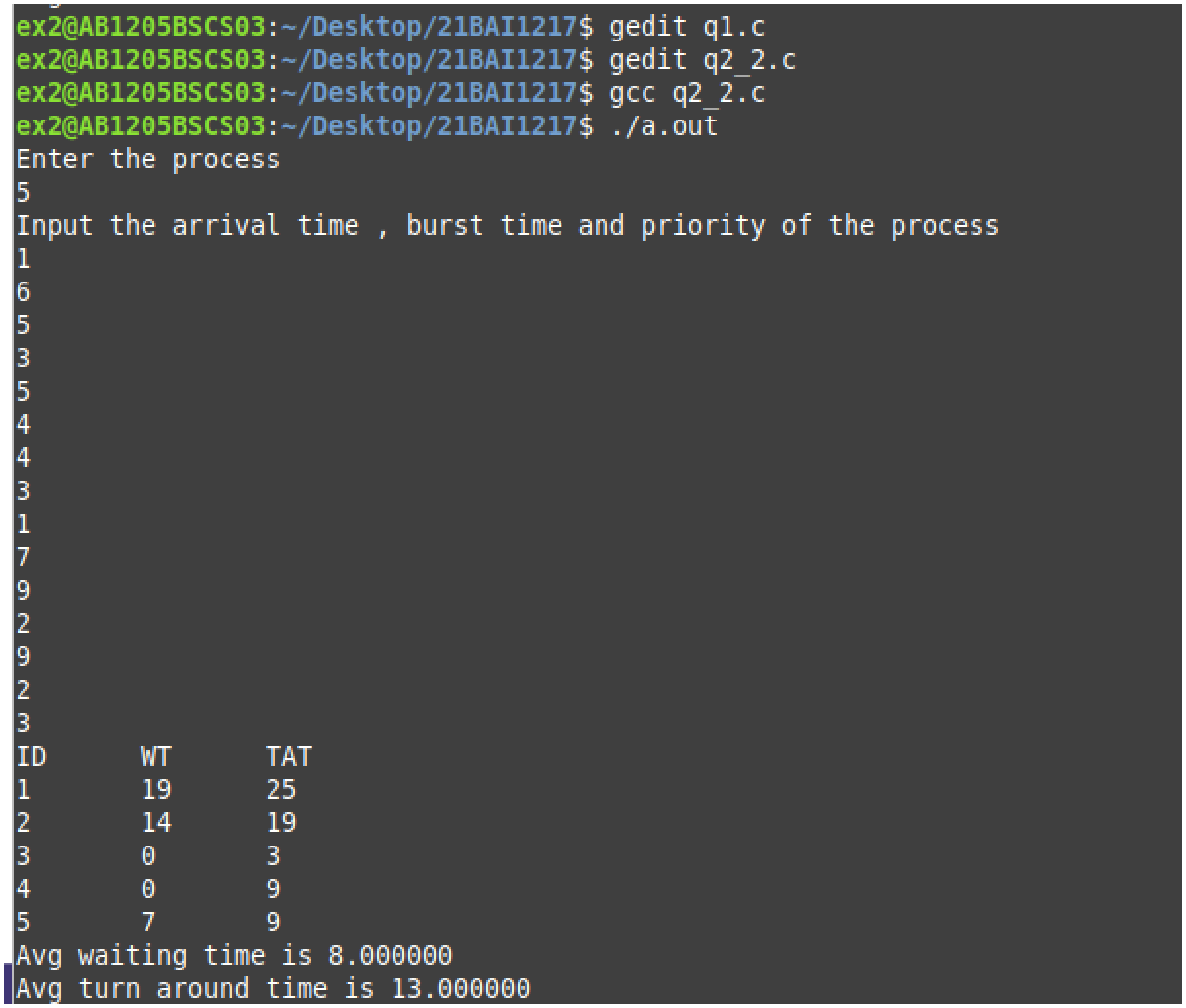
2

3

**CODE**

```
#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,i;
float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
printf("Enter the process\n");
scanf("%d",&n);
printf("Input the arrival time , burst time and priority of the process\n");
for(i=0;i<n;i++)
{
scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);
// adding a duplicate of the burst time
// temporary array for future use
temp[i]=a[i].BT;
}
// initializing burst time
// of a process with maximum
a[9].PT=10000;
for(t=0;count!=n;t++)
{
short_p=9;
for(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 condition on 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 cal
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\tWT\tTAT\n");
for(i=0;i<n;i++)
{
printf("%d\t%d\t%d\n",i+1,a[i].WT,a[i].TAT);
}
printf("Avg waiting time is %f\n",Avg_WT);
printf("Avg turn around time is %f\n",Avg_TAT);
return 0;
}
```

**OUTPUT**



```
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ gedit q1.c
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ gedit q2_2.c
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ gcc q2_2.c
ex2@AB1205BSCS03:~/Desktop/21BAI1217$ ./a.out
Enter the process
5
Input the arrival time , burst time and priority of the process
1
6
5
3
5
4
4
3
1
7
9
2
9
2
3
ID      WT      TAT
1       19      25
2       14      19
3       0       3
4       0       9
5       7       9
Avg waiting time is 8.000000
Avg turn around time is 13.000000
```