KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

OS LAB -6

- Name :HITU RAJ
- Roll no. :2005025
- Branch :CSE

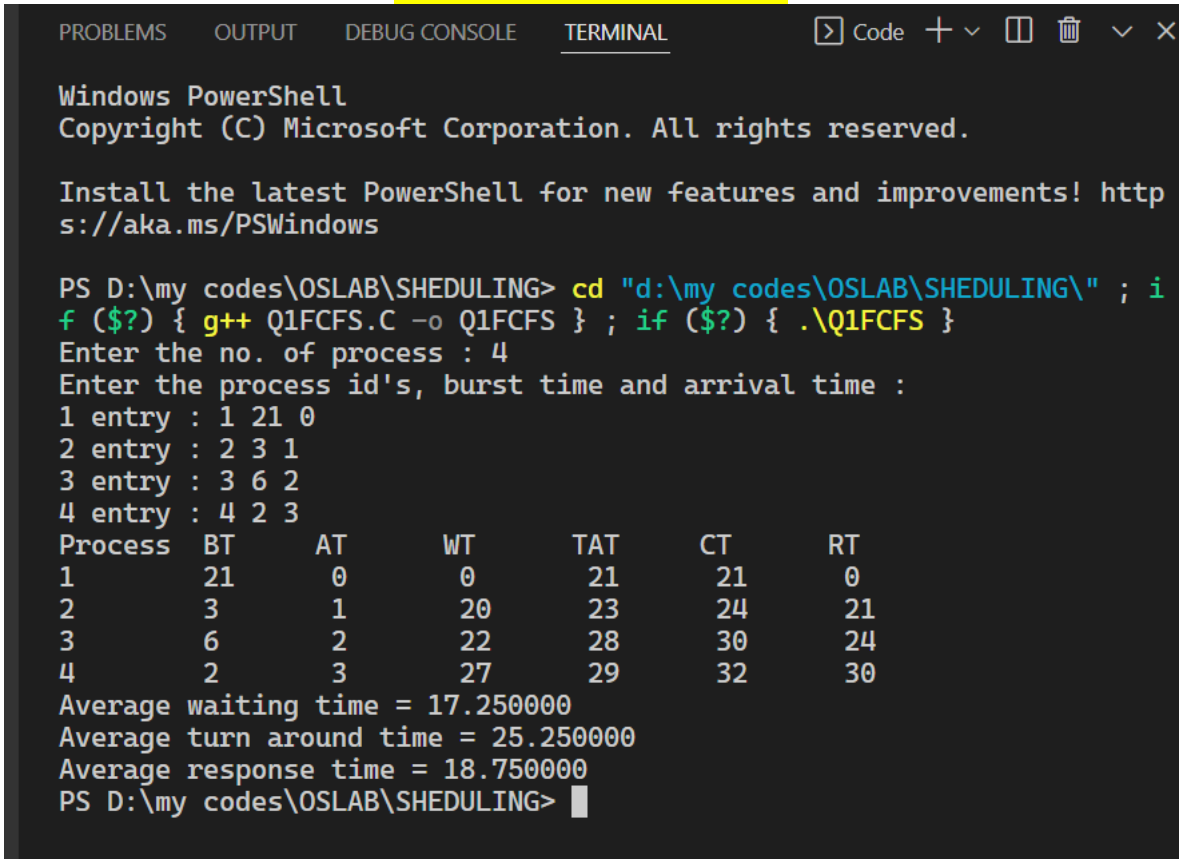
Q1.FIRST COME FIRST SERVE(FCFS)

```
#include <stdio.h>
void findWaitingTime(int processes[], int n, int bt[], int wt[], int at[])
{
    int service_time[n];
    service_time[0] = at[0];
    wt[0] = 0; // calculating waiting time
    for (int i = 1; i < n; i++)
    { // Add burst time of previous processes
        service_time[i] = service_time[i - 1] + bt[i - 1];

        // Find waiting time for current process = sum - at[i]
        wt[i] = service_time[i] - at[i];
        // If waiting time for a process is in negative that means it is already in the
ready queue before CPU becomes idle so its waiting time is 0
        if (wt[i] < 0)
            wt[i] = 0;
    }
```

```c
} // Function to calculate turn around time
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{ // Calculating turnaround time by adding bt[i] + wt[i]
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}
// Function to calculate response time
void findResponseTime(int processes[], int n, int rt[], int wt[], int at[])
{ // Calculating response time by adding at[i] + wt[i]
    for (int i = 0; i < n; i++)
        rt[i] = wt[i] + at[i];
} // Function to calculate average waiting and turn-around times.
void findavgTime(int processes[], int n, int bt[], int at[])
{
    int wt[n], tat[n], rt[n];                       // Function to find waiting time of
all processes
    findWaitingTime(processes, n, bt, wt, at);     // Function to find turn around time
for all processes
    findTurnAroundTime(processes, n, bt, wt, tat); // Function to find response time
for all processes
    findResponseTime(processes, n, rt, wt, at);    // Display processes along with all
details
    printf("Process\t BT \tAT \tWT \tTAT \tCT \tRT\n");
    int total_wt = 0, total_tat = 0, total_rt = 0;
    for (int i = 0; i < n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        total_rt = total_rt + rt[i];
        int compl_time = tat[i] + at[i];
        printf("%d\t %d\t %d\t %d\t %d\t %d\t %d\n", i + 1, bt[i], at[i], wt[i],
tat[i], compl_time, rt[i]);
    }
    printf("Average waiting time = %f\n", (float)total_wt / n);
    printf("Average turn around time = %f\n", (float)total_tat / n);
    printf("Average response time = %f\n", (float)total_rt / n);
}
int main()
{
    int n = 0;
    printf("Enter the no. of process : ");
    scanf("%d", &n); // process id's , Burst time , arrival time of all processes
    int processes[n], burst_time[n], arrival_time[n];
    printf("Enter the process id's, burst time and arrival time : \n");
    for (int i = 0; i < n; i++)
    {
        printf("%d entry : ", i + 1);
        scanf("%d%d%d", &processes[i], &burst_time[i], &arrival_time[i]);
    }
    findavgTime(processes, n, burst_time, arrival_time);
    return 0;
}
```

OUTPUT -1



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! http
s://aka.ms/PSWindows

PS D:\my codes\OSLAB\SHEDULING> cd "d:\my codes\OSLAB\SHEDULING\" ; i
f ($?) { g++ Q1FCFS.C -o Q1FCFS } ; if ($?) { .\Q1FCFS }
Enter the no. of process : 4
Enter the process id's, burst time and arrival time :
1 entry : 1 21 0
2 entry : 2 3 1
3 entry : 3 6 2
4 entry : 4 2 3
Process  BT     AT      WT      TAT     CT      RT
1        21      0       0       21      21      0
2        3       1       20      23      24      21
3        6       2       22      28      30      24
4        2       3       27      29      32      30
Average waiting time = 17.250000
Average turn around time = 25.250000
Average response time = 18.750000
PS D:\my codes\OSLAB\SHEDULING>
```

Q2. SHORTEST JOB FIRST (SJF)

```c
// Question 2 : Shortest Job First(SJF) Scheduling
#include <stdio.h>
int main()
{
    int i, n, p[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, min, k = 0, btime = 0;
    int bt[10], temp, j, at[10], wt[10], tt[10], ta = 0, sum = 0;
    float wavg = 0, tavg = 0, tsum = 0, wsum = 0;
    printf("SJF (NP)\n");
    printf("\nEnter the No. of processes : ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        printf("Enter the burst time & arrival time of %d process : ", i + 1);
        scanf("%d%d", &bt[i], &at[i]);
    }
    // Sorting According to Arrival Time
    // for (i = 0; i < n; i++)
```

```
    // {
    // for (j = 0; j < n; j++)
    // { // if (at[i] < at[j])
    // {
    // temp = p[j];
    // p[j] = p[i];
    // p[i] = temp;
    // temp = at[j];
    // at[j] = at[i];
    // at[i] = temp;
    // temp = bt[j];
    // bt[j] = bt[i];
    // bt[i] = temp;
    // }
    // }
    // }
    /* Arranging the table according to Burst time, Execution time and Arrival Time
Arrival time <= Execution time */
    for (j = 0; j < n; j++)
    {
        btime = btime + bt[j];
        min = bt[1];
        for (i = k; i < n; i++)
        {
            if (btime >= at[i] && bt[i] < min)
            {
                temp = p[k];
                p[k] = p[i];
                p[i] = temp;
                temp = at[k];
                at[k] = at[i];
                at[i] = temp;
                temp = bt[k];
                bt[k] = bt[i];
                bt[i] = temp;
            }
        }
        k++;
    }
    wt[0] = 0;
    for (i = 1; i < n; i++)
    {
        sum = sum + bt[i - 1];
        wt[i] = sum - at[i];
        wsum = wsum + wt[i];
    }
    wavg = (wsum / n);
    for (i = 0; i < n; i++)
    {
        ta = ta + bt[i];
        tt[i] = ta - at[i];
        tsum = tsum + tt[i];
    }
    tavg = (tsum / n);
```

```
    printf("\nProcess\t\tBurst\t\tArrival\t\tWaiting\t\tTurn-around");
    for (i = 0; i < n; i++)
    {
        printf("\n p%d\t\t %d\t\t %d\t\t %d\t\t %d", p[i], bt[i], at[i], wt[i], tt[i]);
    }
    printf("\n\nAVERAGE WAITING TIME : %f", wavg);
    printf("\nAVERAGE TURN AROUND TIME : %f \n", tavg);
    return 0;
}
```

OUTPUT-2

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\my codes\OSLAB\SHEDULING> cd "d:\my codes\OSLAB\SHEDULING\" ; if ($?) { g++ Q2SJF.C -o Q2SJF } ; if ($?) { .\Q2SJF }
SJF (NP)

Enter the No. of processes : 4
Enter the burst time & arrival time of 1 process : 21 0
Enter the burst time & arrival time of 2 process : 3 0
Enter the burst time & arrival time of 3 process : 6 0
Enter the burst time & arrival time of 4 process : 2 0

Process         Burst           Arrival         Waiting         Turn-around
 p4              2               0               0               2
 p2              3               0               2               5
 p3              6               0               5               11
 p1              21              0               11              32

AVERAGE WAITING TIME : 4.500000
AVERAGE TURN AROUND TIME : 12.500000
PS D:\my codes\OSLAB\SHEDULING>
```

Q3-ROUND ROBIN

```
// Question 3 : Round -Robin Scheduling
#include <stdio.h>
struct process
{
    char name;
    int at, bt, wt, tt, rt;
    int completed;
    float ntt;
} 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 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;
            }
        }
    }
}
int main()
{
    int i, j, time = 0, sum_bt = 0, tq;
    int c;
    float avgwt = 0, avgtt = 0;
    printf("Enter no of processes : ");
    scanf("%d", &n);
    for (i = 0, c = 1; i < n; i++, c++)
    {
        p[i].name = c;
```

```
        printf("\nEnter the arrival time and burst time of process %d: ", i + 1);
        scanf("%d%d", &p[i].at, &p[i].bt);
        p[i].rt = p[i].bt;
        p[i].completed = 0;
        sum_bt += p[i].bt;
    }
    printf("\nEnter the time quantum : ");
    scanf("%d", &tq);
    sortByArrival(); // sorting on the basis of arrival time
    enqueue(0);      // enqueue the first
    printf("Process execution order : ");
    for (time = p[0].at; time < sum_bt;) // run until the total burst timereached
    {
        i = dequeue();
        if (p[i].rt <= tq)
        { /* for processes having remaining time with less than or equal to time
quantum */
            time += p[i].rt;
            p[i].rt = 0;
            p[i].completed = 1;
            printf(" %d ", p[i].name);
            p[i].wt = time - p[i].at - p[i].bt;
            p[i].tt = time - p[i].at;
            p[i].ntt = ((float)p[i].tt / p[i].bt);
            for (j = 0; j < n; j++) /*enqueue the processes which have come while
scheduling */
            {
                if (p[j].at <= time && p[j].completed != 1 && isInQueue(j) != 1)
                {
                    enqueue(j);
                }
            }
        }
        else // more than time quantum
        {
            time += tq;
            p[i].rt -= tq;
            printf(" %d ", p[i].name);
            for (j = 0; j < n; j++) /*first enqueue the processes which have come while
scheduling */
            {
                if (p[j].at <= time && p[j].completed != 1 && i != j && isInQueue(j) !=
1)
                {
                    enqueue(j);
                }
            }
            enqueue(i); // then enqueue the uncompleted process
        }
    }
    printf("\nName\tArrival Time\tBurst Time\tWaiting Time\tTurnAround Time\n");
    for (i = 0; i < n; i++)
    {
        avgwt += p[i].wt;
```

```
        avgtt += p[i].tt;
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t", p[i].name, p[i].at, p[i].bt,
p[i].wt, p[i].tt);
    }
    printf("\n\nAverage waiting time : %f\n", avgwt / n);
    printf("Average turn around time : %f\n", avgtt / n);
    return 0;
}
```

OUTPUT -3

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                  Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\my codes\OSLAB\SHEDULING> cd "d:\my codes\OSLAB\SHEDULING\" ; if ($?) { g++ Q3ROUNDROBIN.C -o Q3ROUNDROBIN } ; if ($?) { .\Q3ROUNDROBIN }
Enter no of processes : 4

Enter the arrival time and burst time of process 1: 0 21

Enter the arrival time and burst time of process 2: 0 3

Enter the arrival time and burst time of process 3: 0 6

Enter the arrival time and burst time of process 4:  0 2

Enter the time quantum : 5
Process execution order :  1  2  3  4  1  3  1  1  1
Name    Arrival Time    Burst Time      Waiting Time    TurnAround Time

1               0               21              11              32
2               0               3               5               8
3               0               6               15              21
4               0               2               13              15

Average waiting time : 11.000000
Average turn around time : 19.000000
PS D:\my codes\OSLAB\SHEDULING>
```

Q4A PRIORITY SCHEDULING (PREMPTIVE)

```
//   Question 4 (A): Priority Scheduling Pre -emptive
#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 : ");
    scanf("%d", &n);
    printf("Enter the burst time, priority and arrival time of the process\n");
    printf("BT PT AT\n");
    for (int i = 0; i < n; i++)

        scanf("%d%d%d", &a[i].BT, &a[i].PT, &a[i].AT); // copying the burst time in
        // a temp array fot futher use
        temp[i] = a[i].BT;
    }
    // we initialize the priority
    // 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("\nProcess.ID\tWaiting Time\tTurn Around Time\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t\t%d\t\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;
}
```

OUTPUT 4(A)

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\my codes\OSLAB\SHEDULING> cd "d:\my codes\OSLAB\SHEDULING\" ; if ($?) { g++ Q4APRIORITY_PREM.C -o Q4APRIORITY_PREM
} ; if ($?) { .\Q4APRIORITY_PREM }
Enter the number of the process : 4
Enter the burst time, priority and arrival time of the process
BT PT AT
21 2 0
3 1 5
6 4 7
2 3 8

Process.ID      Waiting Time    Turn Around Time
1               3               24
2               0               3
3               19              25
4               16              18
Avg waiting time of the process is 9.500000
Avg turn around time of the process is 17.500000
PS D:\my codes\OSLAB\SHEDULING>
```

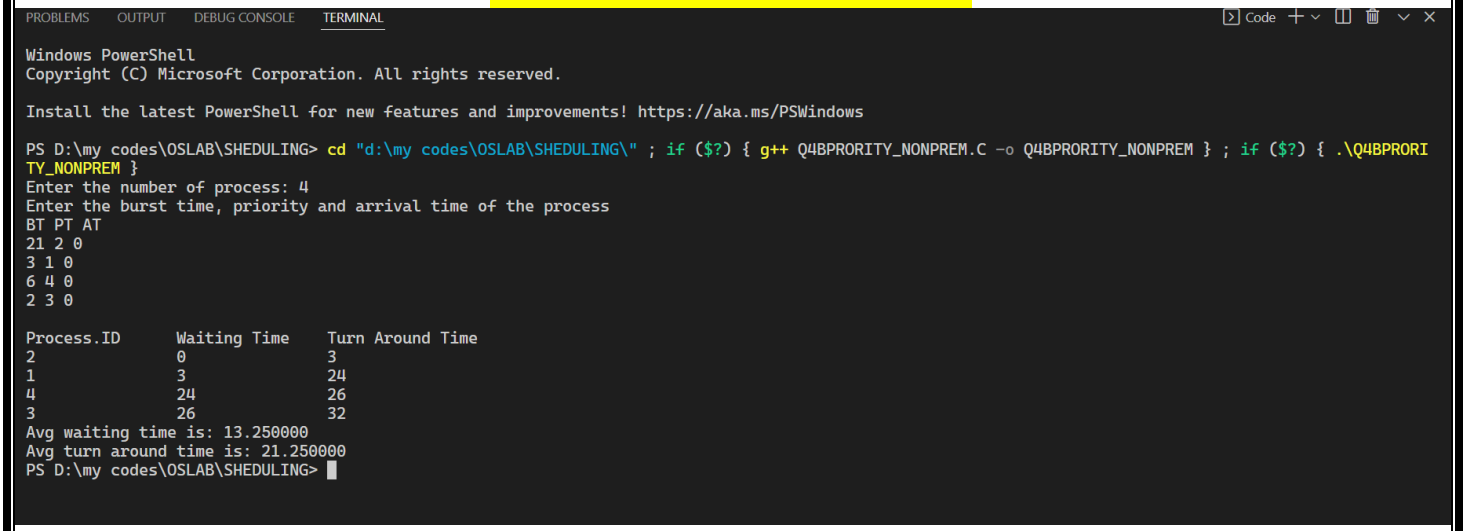
Q4B PRIORITY SCHEDULING (NON-PREMPTIVE)

```c
// (4B)Non Pre -emptive
#include<stdio.h>
 struct process
{
    int id, WT, AT, BT, TAT, PR;
};
struct process a[10]; // function for swapping
void swap(int *b, int *c)
{
    int tem;
    tem = *c;
    *c = *b;
    *b = tem;
}
int main()
{
    int n, check_ar = 0;
    int Cmp_time = 0;
    float Total_WT = 0, Total_TAT = 0, Avg_WT, Avg_TAT;
    printf("Enter the number of process: ");
    scanf("%d", &n);
    printf("Enter the burst time, priority and arrival time of the process\n");
    printf("BT PT AT\n");
    for (int i = 0; i < n; i++)
```

```
    {
        scanf("%d%d%d", &a[i].BT, &a[i].PR, &a[i].AT);
        a[i].id = i + 1; // here we are checking that arrival time // of the process
are same or different
        if (i == 0)
            check_ar = a[i].AT;
        if (check_ar != a[i].AT)
            check_ar = 1;
    } // if process are arrived at the different time // then sort the process on the
basis of AT
    if (check_ar != 0)
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n - i - 1; j++)
            {
                if (a[j].AT > a[j + 1].AT)
                {
                    swap(&a[j].id, &a[j + 1].id);
                    swap(&a[j].AT, &a[j + 1].AT);
                    swap(&a[j].BT, &a[j + 1].BT);
                    swap(&a[j].PR, &a[j + 1].PR);
                }
            }
        }
    } // logic of Priority scheduling ( non preemptive) algo // if all the process are
arrived at different time
    if (check_ar != 0)
    {
        a[0].WT = a[0].AT;
        a[0].TAT = a[0].BT - a[0].AT; // cmp_time for completion time
        Cmp_time = a[0].TAT;
        Total_WT = Total_WT + a[0].WT;
        Total_TAT = Total_TAT + a[0].TAT;
        for (int i = 1; i < n; i++)
        {
            int min = a[i].PR;
            for (int j = i + 1; j < n; j++)
            {
                if (min > a[j].PR && a[j].AT <= Cmp_time)
                {
                    min = a[j].PR;
                    swap(&a[i].id, &a[j].id);
                    swap(&a[i].AT, &a[j].AT);
                    swap(&a[i].BT, &a[j].BT);
                    swap(&a[i].PR, &a[j].PR);
                }
            }
            a[i].WT = Cmp_time - a[i].AT;
            Total_WT = Total_WT + a[i].WT; // completion time of the process
            Cmp_time = Cmp_time + a[i].BT; // Turn Around Time of the process // compl
-Arival
            a[i].TAT = Cmp_time - a[i].AT;
            Total_TAT = Total_TAT + a[i].TAT;
```

```
        }
    }
    // if all the process are arrived at same time
    else
    {
        for (int i = 0; i < n; i++)
        {
            int min = a[i].PR;
            for (int j = i + 1; j < n; j++)
            {
                if (min > a[j].PR && a[j].AT <= Cmp_time)
                {
                    min = a[j].PR;
                    swap(&a[i].id, &a[j].id);
                    swap(&a[i].AT, &a[j].AT);
                    swap(&a[i].BT, &a[j].BT);
                    swap(&a[i].PR, &a[j].PR);
                }
            }
            a[i].WT = Cmp_time - a[i].AT;  // completion time of the process
            Cmp_time = Cmp_time + a[i].BT; // Turn Around Time of the process
                                           // compl -Arrival
            a[i].TAT = Cmp_time - a[i].AT;
            Total_WT = Total_WT + a[i].WT;
            Total_TAT = Total_TAT + a[i].TAT;
        }
    }
    Avg_WT = Total_WT / n;
    Avg_TAT = Total_TAT / n;
    // Printing of the results
    printf("\nProcess.ID\tWaiting Time\tTurn Around Time\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t\t%d\t\t%d\n", a[i].id, 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 4(B)


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS D:\my codes\OSLAB\SHEDULING> cd "d:\my codes\OSLAB\SHEDULING\" ; if ($?) { g++ Q4BPRORITY_NONPREM.C -o Q4BPRORITY_NONPREM } ; if ($?) { .\Q4BPRORITY_NONPREM }
Enter the number of process: 4
Enter the burst time, priority and arrival time of the process
BT PT AT
21 2 0
3 1 0
6 4 0
2 3 0
Process.ID Waiting Time Turn Around Time
2 0 3
1 3 24
4 24 26
3 26 32
Avg waiting time is: 13.250000
Avg turn around time is: 21.250000
PS D:\my codes\OSLAB\SHEDULING>