ARTI BHATIA

CS5A, 16

**CPU SCHEDULING**

Q- Simulate the following non-preemptive CPU scheduling algorithms to find turnaround

time and waiting time.

1. FCFS
2. SJN
3. Round Robin (pre-emptive)
4. Priority

**First Come First Serve**

Program:

#include <stdio.h>

int n, at[10], bt[10], tt[10], wt[10], ct[10], pno[10];

int total\_tt = 0, total\_wt = 0;

float avg\_tt, avg\_wt;

void swap(int \*x, int \*y)

{

    int temp = \*x;

    \*x = \*y;

    \*y = temp;

}

void sort()

{

    for (int i = 0; i < n; i++)

    {

        int shortest = i;

        for (int j = i + 1; j < n; j++)

        {

            if (at[j] < at[shortest])

            {

                shortest = j;

            }

        }

        swap(&at[i], &at[shortest]);

        swap(&bt[i], &bt[shortest]);

        swap(&pno[i], &pno[shortest]);

    }

}

void fcfs()

{

    int exitTime = at[0];

    for (int i = 0; i < n; i++)

    {

        ct[i] = exitTime + bt[i];

        tt[i] = ct[i] - at[i];

        wt[i] = tt[i] - bt[i];

        exitTime = ct[i];

        total\_tt += tt[i];

        total\_wt += wt[i];

    }

}

void show()

{

    printf("\n\nPno\t\tAT\t\tBT\t\tCT\t\tTT\t\tWT");

    for (int i = 0; i < n; i++)

    {

        printf("\nP%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d", pno[i], at[i], bt[i], ct[i], tt[i], wt[i]);

    }

    avg\_tt = (float)total\_tt / n;

    avg\_wt = (float)total\_wt / n;

    printf("\n\nAverage TT : %.3f", avg\_tt);

    printf("\nAverage WT : %.3f", avg\_wt);

}

void main()

{

    printf("\nNumber of processes: ");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        printf("arrival time: ");

        scanf("%d", &at[i]);

        printf("burst time: ");

        scanf("%d", &bt[i]);

    }

    for (int i = 0; i < n; i++)

        pno[i] = i + 1;

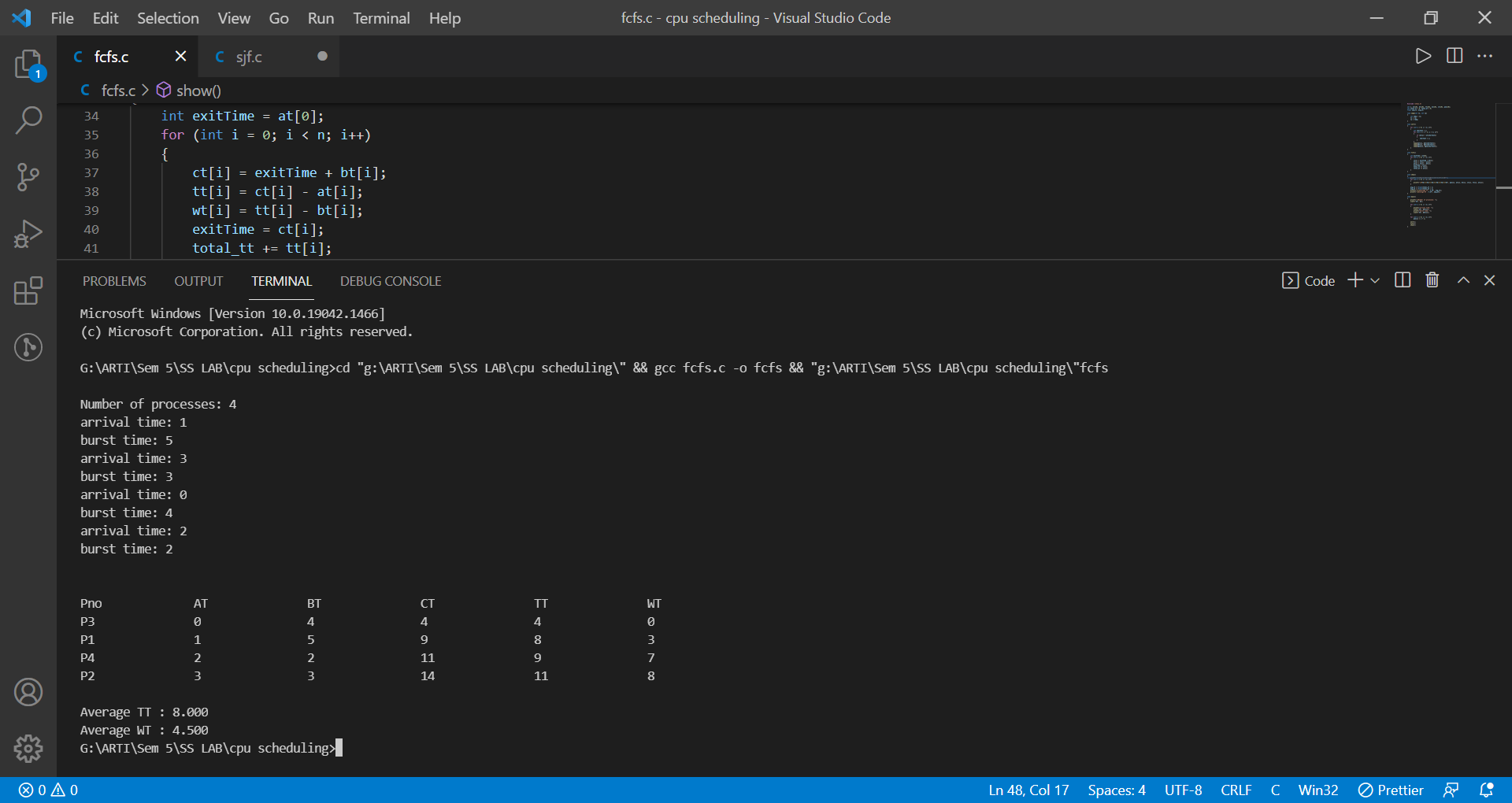
    sort();

    fcfs();

    show();

}

Output:



**Shortest Job Next**

Program:

#include <stdio.h>

#include <limits.h>

int n, at[10], bt[10], tt[10], wt[10], ct[10], pno[10], visited[10];

int total\_tt = 0, total\_wt = 0;

float avg\_tt, avg\_wt;

void sjn()

{

    int count = 0, currTime = 0;

    while (count != n)

    {

        int pid = -1, min\_bt = INT\_MAX;

        for (int i = 0; i < n; i++)

        {

            if (at[i] <= currTime && visited[i] == 0)

            {

                if (bt[i] < min\_bt)

                {

                    min\_bt = bt[i];

                    pid = i;

                }

                else if (bt[i] == min\_bt)

                {

                    if (at[i] < at[pid])

                        pid = i;

                }

            }

        }

        if (pid != -1)

        {

            currTime += bt[pid];

            ct[pid] = currTime;

            tt[pid] = ct[pid] - at[pid];

            wt[pid] = tt[pid] - bt[pid];

            visited[pid] = 1;

            total\_tt += tt[pid];

            total\_wt += wt[pid];

            count++;

        }

        else

            currTime++;

    }

}

void show()

{

    printf("\nPno\t\tAT\t\tBT\t\tCT\t\tTT\t\tWT");

    for (int i = 0; i < n; i++)

    {

        printf("\nP%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d", pno[i], at[i], bt[i], ct[i], tt[i], wt[i]);

    }

    avg\_tt = (float)total\_tt / n;

    avg\_wt = (float)total\_wt / n;

    printf("\n\nAverage TT : %.3f", avg\_tt);

    printf("\nAverage WT : %.3f", avg\_wt);

}

void main()

{

    printf("\nNumber of processes: ");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        printf("arrival time: ");

        scanf("%d", &at[i]);

        printf("burst time: ");

        scanf("%d", &bt[i]);

    }

    for (int i = 0; i < n; i++)

    {

        pno[i] = i + 1;

        visited[i] = 0;

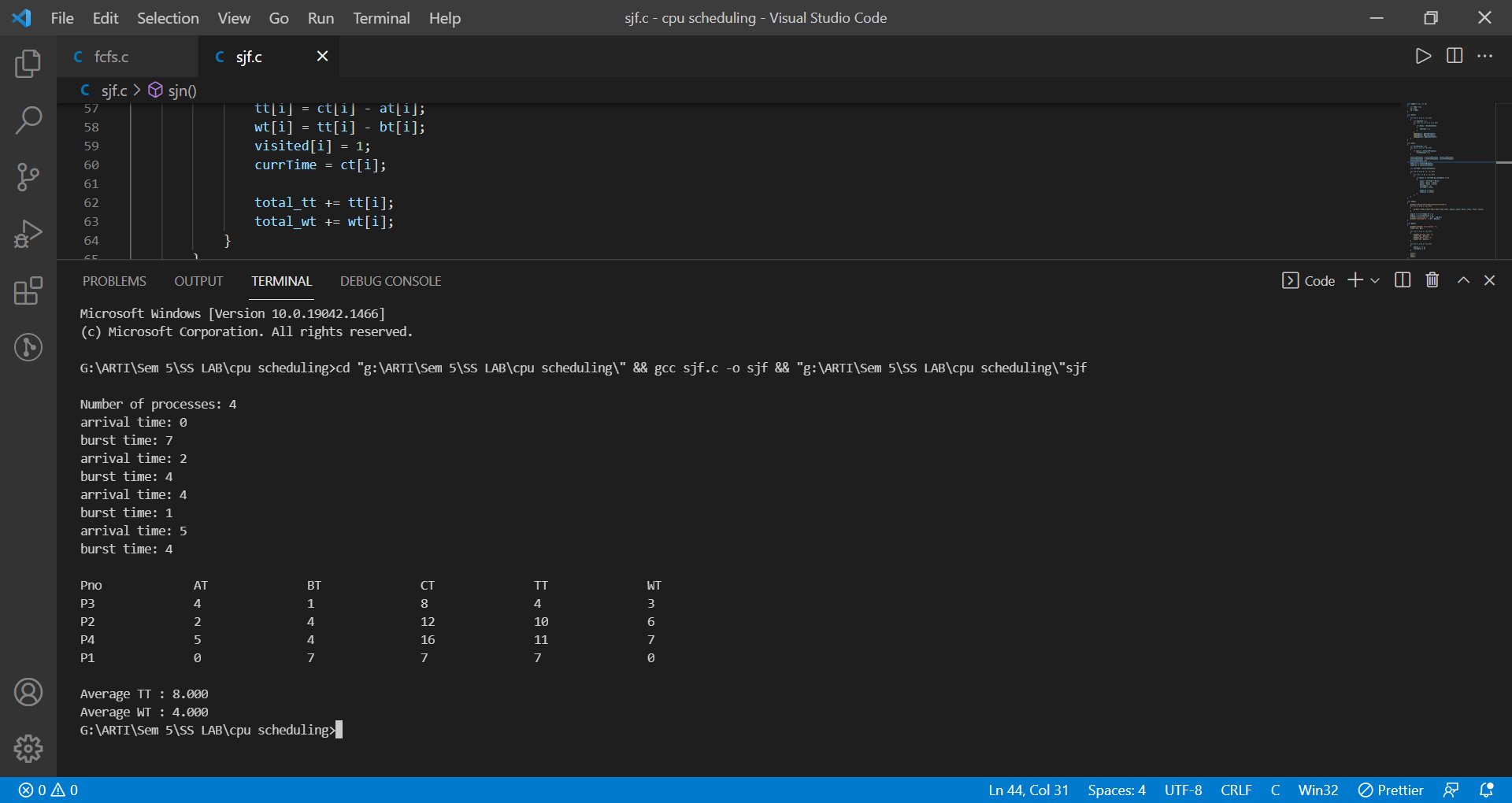
    }

    sjn();

    show();

}

Output:



**Round Robin (pre- emptive)**

Program:

#include <stdio.h>

int n, at[10], bt[10], tt[10], wt[10], rt[10], ct[10], pno[10], completed[10], temp\_bt[10];

int total\_tt = 0, total\_wt = 0;

float avg\_tt, avg\_wt;

const int timeQuantum = 2;

void swap(int \*x, int \*y)

{

    int temp = \*x;

    \*x = \*y;

    \*y = temp;

}

void sort()

{

    for (int i = 0; i < n; i++)

    {

        int shortest = i;

        for (int j = i + 1; j < n; j++)

        {

            if (at[j] < at[shortest])

            {

                shortest = j;

            }

        }

        swap(&at[i], &at[shortest]);

        swap(&bt[i], &bt[shortest]);

        swap(&pno[i], &pno[shortest]);

    }

}

void roundRobin()

{

    int currTime = at[0];

    int count = 0, i = 0;

    for (int i = 0; i < n; i++)

        temp\_bt[i] = bt[i];

    while (count != n)

    {

        if (at[i] > currTime)

            currTime = at[i];

        if (temp\_bt[i] == bt[i])

            rt[i] = currTime - at[i];

        if (temp\_bt[i] <= timeQuantum && temp\_bt[i] > 0)

        {

            currTime += temp\_bt[i];

            temp\_bt[i] = 0;

            ct[i] = currTime;

            tt[i] = ct[i] - at[i];

            wt[i] = tt[i] - bt[i];

            total\_tt += tt[i];

            total\_wt += wt[i];

            count++;

        }

        else if (temp\_bt[i] > 0)

        {

            temp\_bt[i] -= timeQuantum;

            currTime += timeQuantum;

        }

        if (i == n - 1)

            i = 0;

        else

            i++;

    }

}

void show()

{

    printf("\nPno\t\tAT\t\tBT\t\tCT\t\tTT\t\tWT\t\tResponse Time");

    for (int i = 0; i < n; i++)

    {

        printf("\nP%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d", pno[i], at[i], bt[i], ct[i], tt[i], wt[i], rt[i]);

    }

    avg\_tt = (float)total\_tt / n;

    avg\_wt = (float)total\_wt / n;

    printf("\n\nAverage TT : %.3f", avg\_tt);

    printf("\nAverage WT : %.3f", avg\_wt);

}

void main()

{

    printf("\nNumber of processes: ");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        printf("arrival time: ");

        scanf("%d", &at[i]);

        printf("burst time: ");

        scanf("%d", &bt[i]);

    }

    for (int i = 0; i < n; i++)

    {

        pno[i] = i + 1;

        completed[i] = 0;

    }

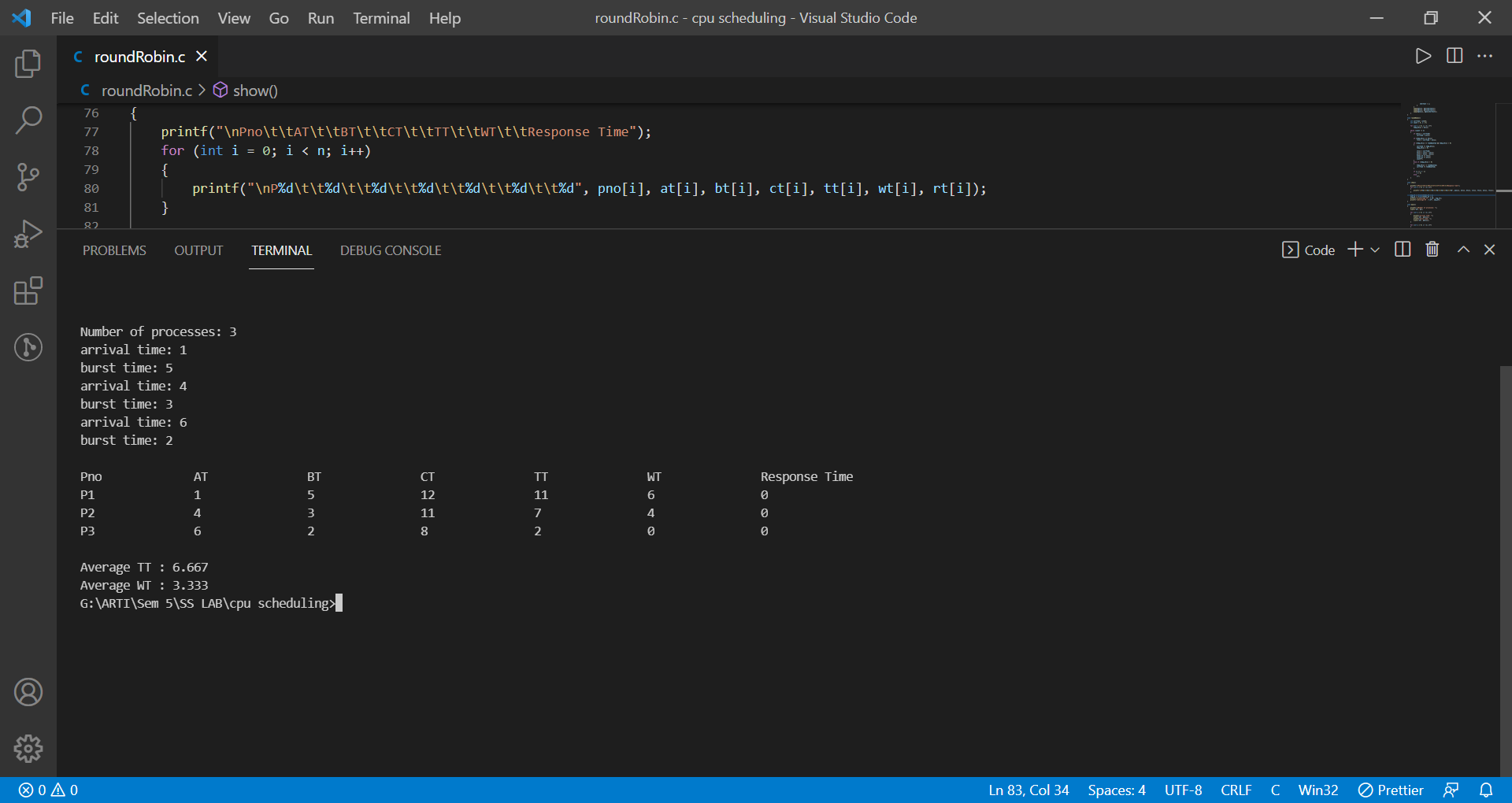
    sort();

    roundRobin();

    show();

}

Output:



**Priority**

Program:

#include <stdio.h>

#include <limits.h>

int n, at[10], bt[10], priority[10], tt[10], wt[10], ct[10], pno[10], visited[10];

int total\_tt = 0, total\_wt = 0;

float avg\_tt, avg\_wt;

void priorityScheduling()

{

    int count = 0, currTime = 0;

    while (count != n)

    {

        int pid = -1, min\_priority = INT\_MAX;

        for (int i = 0; i < n; i++)

        {

            if (at[i] <= currTime && visited[i] == 0)

            {

                if (priority[i] < min\_priority)

                {

                    min\_priority = priority[i];

                    pid = i;

                }

                else if (priority[i] == min\_priority)

                {

                    if (at[i] < at[pid])

                        pid = i;

                }

            }

        }

        if (pid != -1)

        {

            currTime += bt[pid];

            ct[pid] = currTime;

            tt[pid] = ct[pid] - at[pid];

            wt[pid] = tt[pid] - bt[pid];

            visited[pid] = 1;

            total\_tt += tt[pid];

            total\_wt += wt[pid];

            count++;

        }

        else

            currTime++;

    }

}

void show()

{

    printf("\nPno\t\tPriority\t\tAT\t\tBT\t\tCT\t\tTT\t\tWT");

    for (int i = 0; i < n; i++)

    {

        printf("\nP%d\t\t%d\t\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d", pno[i], priority[i], at[i], bt[i], ct[i], tt[i], wt[i]);

    }

    avg\_tt = (float)total\_tt / n;

    avg\_wt = (float)total\_wt / n;

    printf("\n\nAverage TT : %.3f", avg\_tt);

    printf("\nAverage WT : %.3f", avg\_wt);

}

void main()

{

    printf("\nNumber of processes: ");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        printf("priority: ");

        scanf("%d", &priority[i]);

        printf("arrival time: ");

        scanf("%d", &at[i]);

        printf("burst time: ");

        scanf("%d", &bt[i]);

    }

    for (int i = 0; i < n; i++)

    {

        pno[i] = i + 1;

        visited[i] = 0;

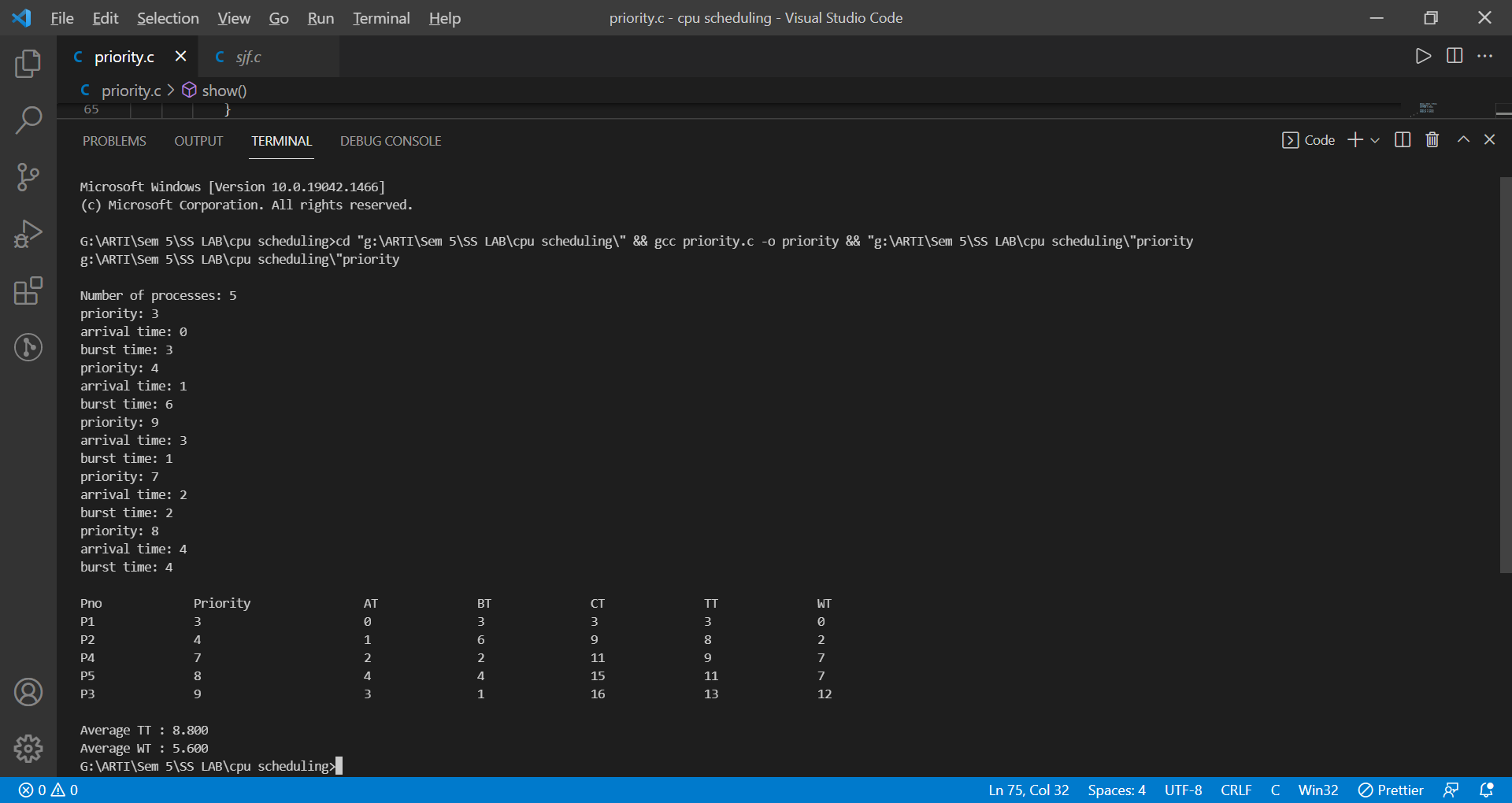
    }

    priorityScheduling();

    show();

}

Output:



**ALGORITHM:**

Algorithm:

