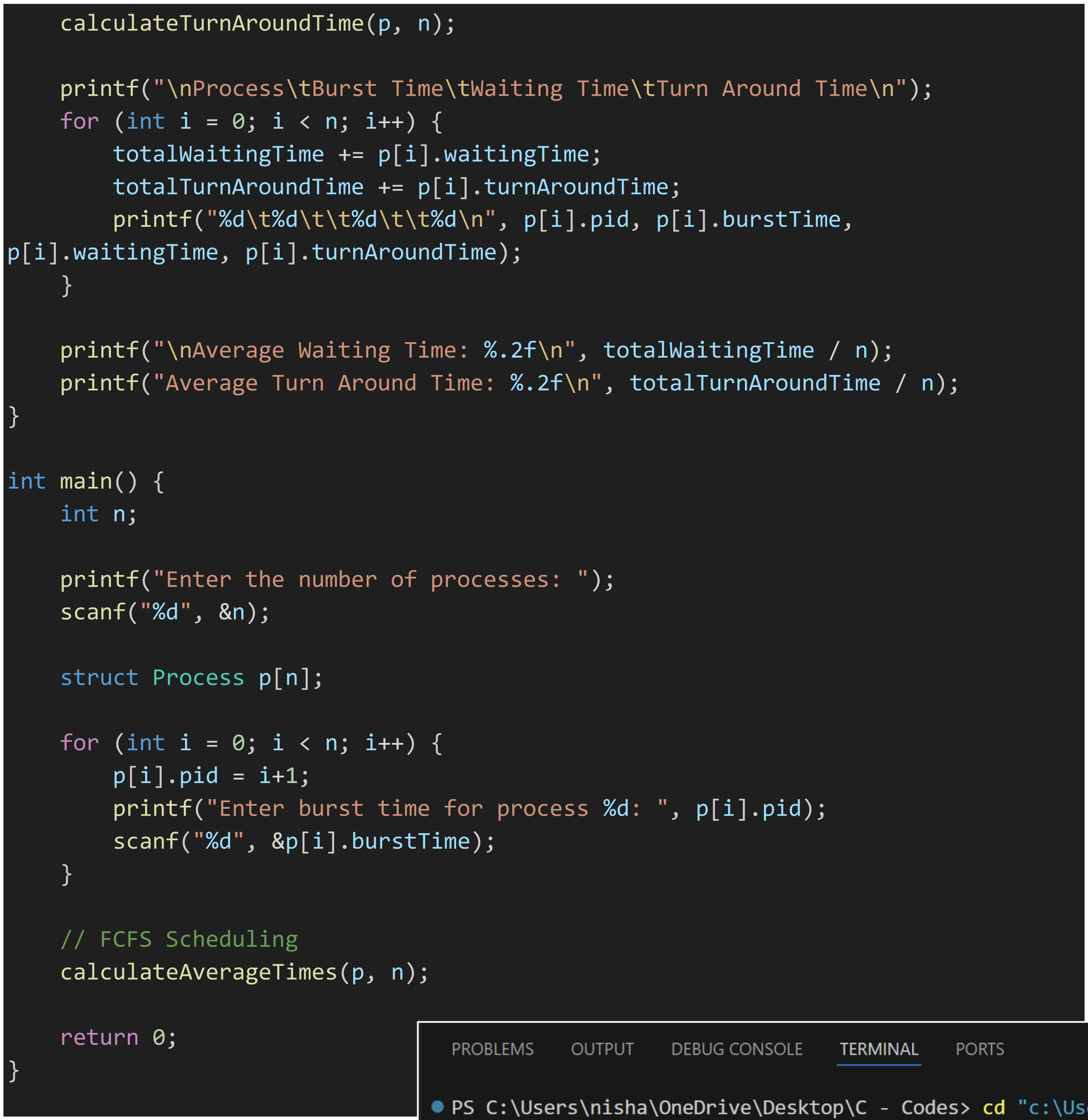
1. Implement the following CPU Scheduling algorithm using C programming.

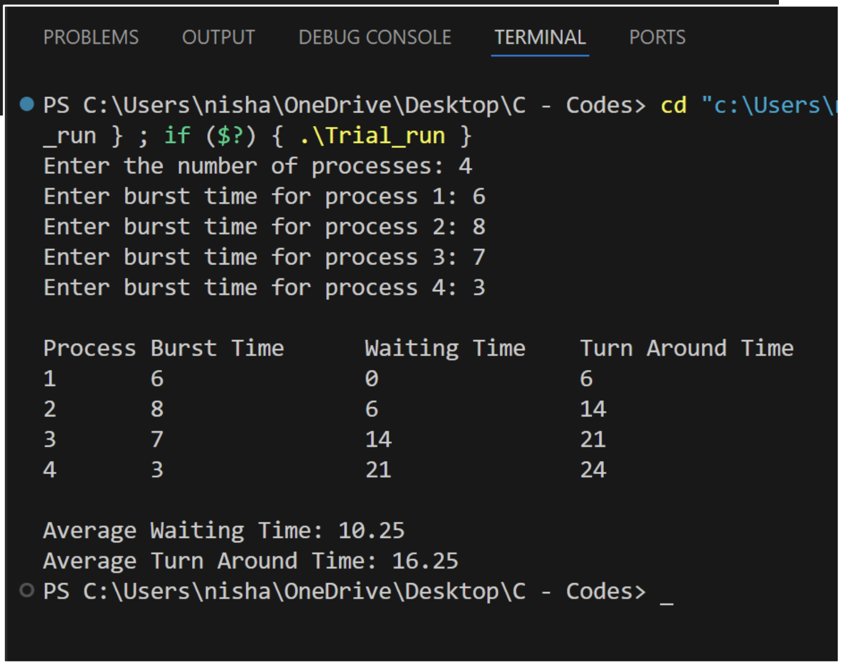
   - First Come First Serve
   - Shortest Job First

FCFS (First come first serve) :-

```c
#include <stdio.h>

struct Process {
    int pid;        // Process ID
    int burstTime; // Burst Time of the process
    int waitingTime; // Waiting Time of the process
    int turnAroundTime; // Turn Around Time of the process
};

// Function to calculate waiting time for each process
void calculateWaitingTime(struct Process p[], int n) {
    p[0].waitingTime = 0; // First process has no waiting time

    for (int i = 1; i < n; i++) {
        p[i].waitingTime = p[i-1].waitingTime + p[i-1].burstTime;
    }
}

// Function to calculate turn around time for each process
void calculateTurnAroundTime(struct Process p[], int n) {
    for (int i = 0; i < n; i++) {
        p[i].turnAroundTime = p[i].waitingTime + p[i].burstTime;
    }
}

// Function to calculate average waiting and turn around times
void calculateAverageTimes(struct Process p[], int n) {
    float totalWaitingTime = 0, totalTurnAroundTime = 0;

    calculateWaitingTime(p, n);
```



```c
    calculateTurnAroundTime(p, n);

    printf("\nProcess\tBurst Time\tWaiting Time\tTurn Around Time\n");
    for (int i = 0; i < n; i++) {
        totalWaitingTime += p[i].waitingTime;
        totalTurnAroundTime += p[i].turnAroundTime;
        printf("%d\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burstTime,
p[i].waitingTime, p[i].turnAroundTime);
    }

    printf("\nAverage Waiting Time: %.2f\n", totalWaitingTime / n);
    printf("Average Turn Around Time: %.2f\n", totalTurnAroundTime / n);
}

int main() {
    int n;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process p[n];

    for (int i = 0; i < n; i++) {
        p[i].pid = i+1;
        printf("Enter burst time for process %d: ", p[i].pid);
        scanf("%d", &p[i].burstTime);
    }

    // FCFS Scheduling
    calculateAverageTimes(p, n);

    return 0;
}
```

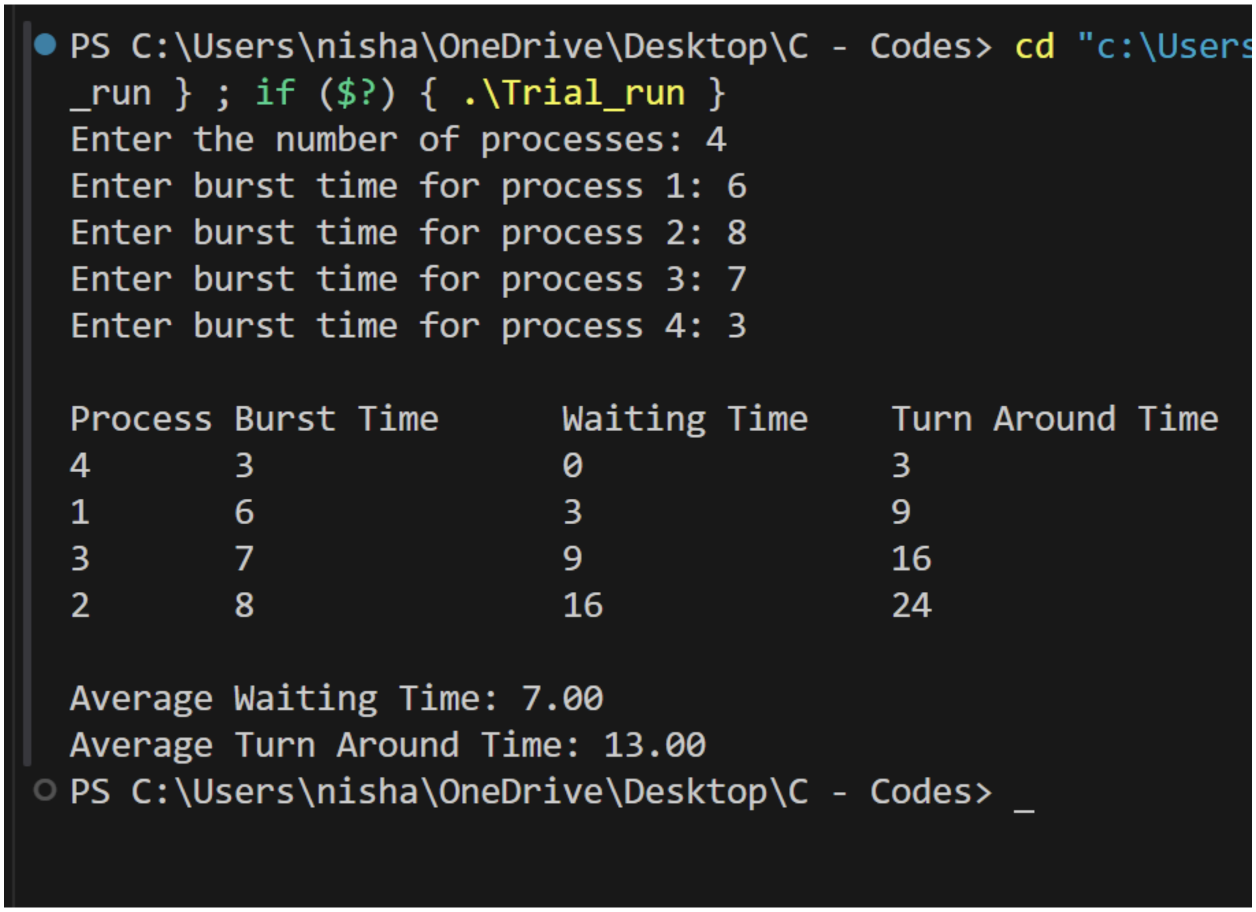


```
PS C:\Users\nisha\OneDrive\Desktop\C - Codes> cd "c:\Users\
_run } ; if ($?) { .\Trial_run }
Enter the number of processes: 4
Enter burst time for process 1: 6
Enter burst time for process 2: 8
Enter burst time for process 3: 7
Enter burst time for process 4: 3

Process Burst Time      Waiting Time    Turn Around Time
1       6               0               6
2       8               6               14
3       7               14              21
4       3               21              24

Average Waiting Time: 10.25
Average Turn Around Time: 16.25
PS C:\Users\nisha\OneDrive\Desktop\C - Codes> _
```

SJF (shortest jobs first) : -

```c
#include <stdio.h>

struct Process {
    int pid;       // Process ID
    int burstTime; // Burst Time of the process
    int waitingTime; // Waiting Time of the process
    int turnAroundTime; // Turn Around Time of the process
};

// Function to swap two processes (used for sorting)
void swap(struct Process *a, struct Process *b) {
    struct Process temp = *a;
    *a = *b;
    *b = temp;
}

// Function to sort the processes according to burst time
void sortByBurstTime(struct Process p[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].burstTime > p[j+1].burstTime) {
                swap(&p[j], &p[j+1]);
            }
        }
    }
}

// Function to calculate waiting time for each process
void calculateWaitingTime(struct Process p[], int n) {
    p[0].waitingTime = 0; // First process has no waiting time

    for (int i = 1; i < n; i++) {
        p[i].waitingTime = p[i-1].waitingTime + p[i-1].burstTime;
    }
}

// Function to calculate turn around time for each process
void calculateTurnAroundTime(struct Process p[], int n) {
    for (int i = 0; i < n; i++) {
        p[i].turnAroundTime = p[i].waitingTime + p[i].burstTime;
    }
}

// Function to calculate average waiting and turn around times
void calculateAverageTimes(struct Process p[], int n) {
    float totalWaitingTime = 0, totalTurnAroundTime = 0;

    calculateWaitingTime(p, n);
    calculateTurnAroundTime(p, n);

    printf("\nProcess\tBurst Time\tWaiting Time\tTurn Around Time\n");
    for (int i = 0; i < n; i++) {
        totalWaitingTime += p[i].waitingTime;
        totalTurnAroundTime += p[i].turnAroundTime;
        printf("%d\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burstTime,
p[i].waitingTime, p[i].turnAroundTime);
    }

    printf("\nAverage Waiting Time: %.2f\n", totalWaitingTime / n);
    printf("Average Turn Around Time: %.2f\n", totalTurnAroundTime / n);
}

int main() {
    int n;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process p[n];

    for (int i = 0; i < n; i++) {
        p[i].pid = i+1;
        printf("Enter burst time for process %d: ", p[i].pid);
        scanf("%d", &p[i].burstTime);
    }

    // Sort processes by burst time for SJF Scheduling
    sortByBurstTime(p, n);

    // SJF Scheduling
    calculateAverageTimes(p, n);

    return 0;
}
```

AP22110010245



```
PS C:\Users\nisha\OneDrive\Desktop\C - Codes> cd "c:\Users
_run } ; if ($?) { .\Trial_run }
Enter the number of processes: 4
Enter burst time for process 1: 6
Enter burst time for process 2: 8
Enter burst time for process 3: 7
Enter burst time for process 4: 3

Process Burst Time      Waiting Time    Turn Around Time
4       3               0               3
1       6               3               9
3       7               9               16
2       8               16              24

Average Waiting Time: 7.00
Average Turn Around Time: 13.00
PS C:\Users\nisha\OneDrive\Desktop\C - Codes> _
```