LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

presented by:-
Name of student: Ayush kumar
Registration number: 12214903

Degree: "B.Tech Computer Science And Engineering"

https://github.com/ayushgithuboro/cse-316

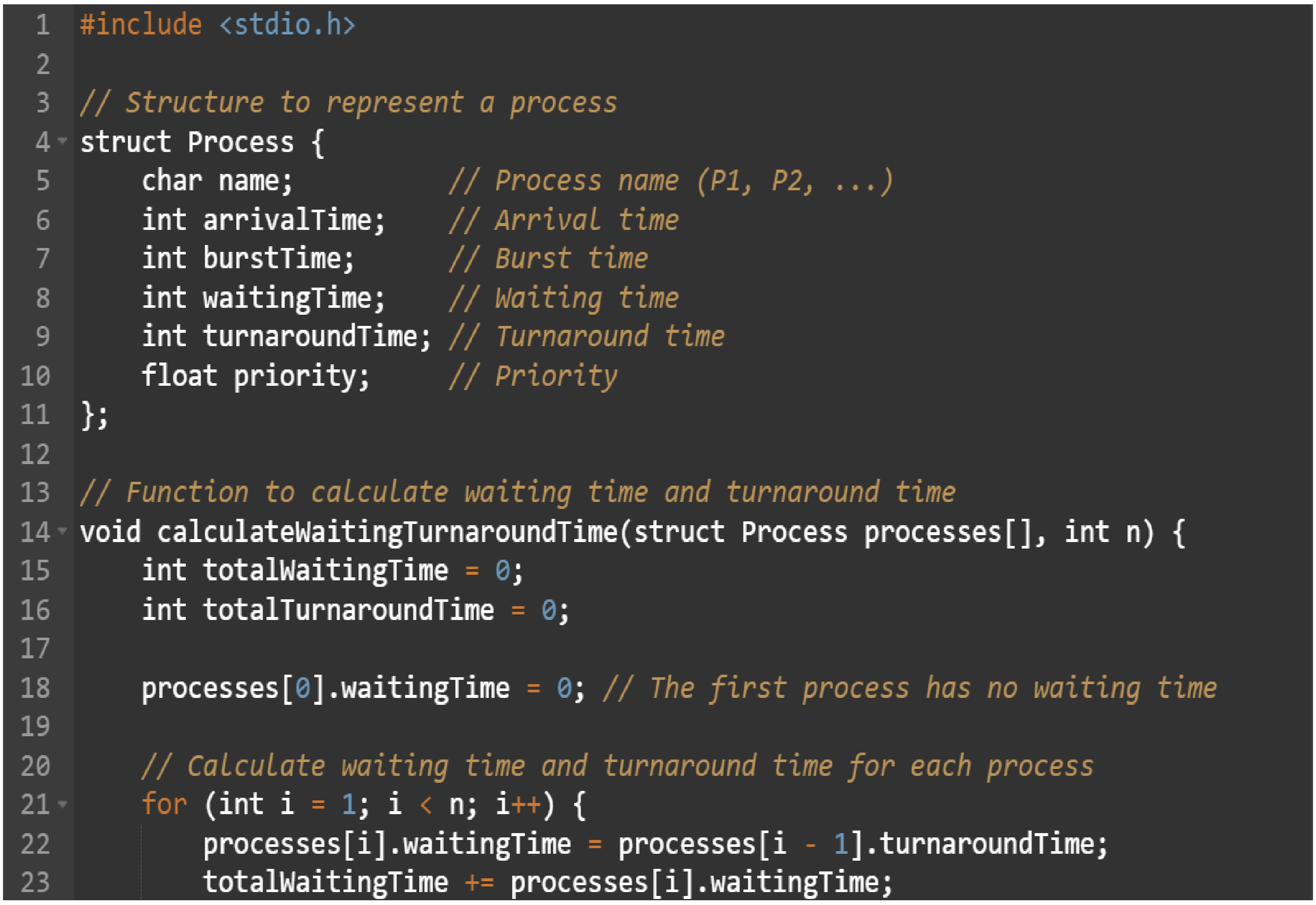
Ques. . Consider a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times. The priority can be computed as :

Priority = 1+ Waiting time / Estimated run time

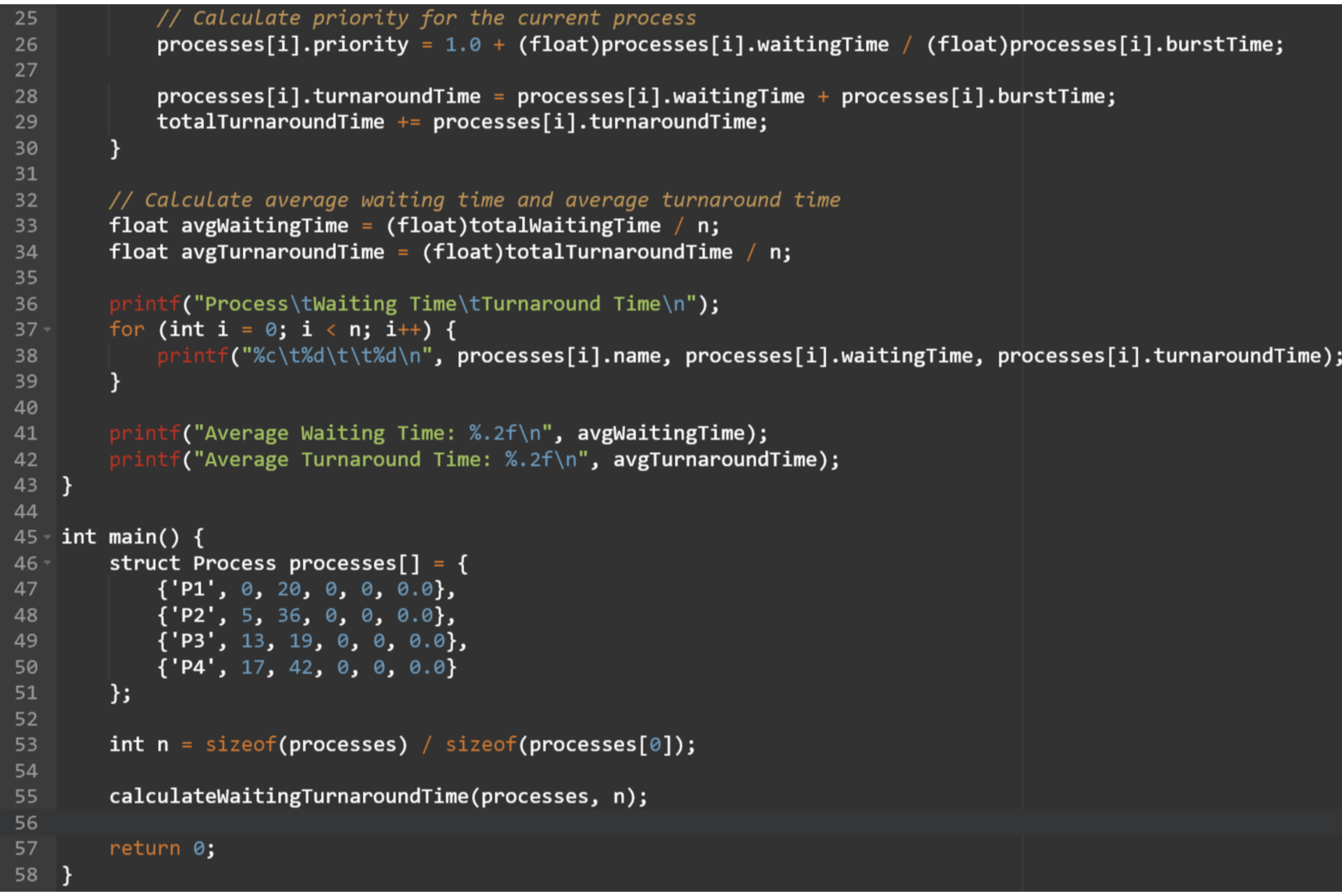
Using the data given below compute the waiting time and turnaround time for each process and average waiting time and average turnaround time.

Process	Arrival time	Burst time
P_1	0	20
P_2	5	36
P_3	13	19
P_4	17	42

Ans: This C program calculates the waiting time and turnaround time for the given processes using the priority formula and then calculates the average waiting time and average turnaround time.

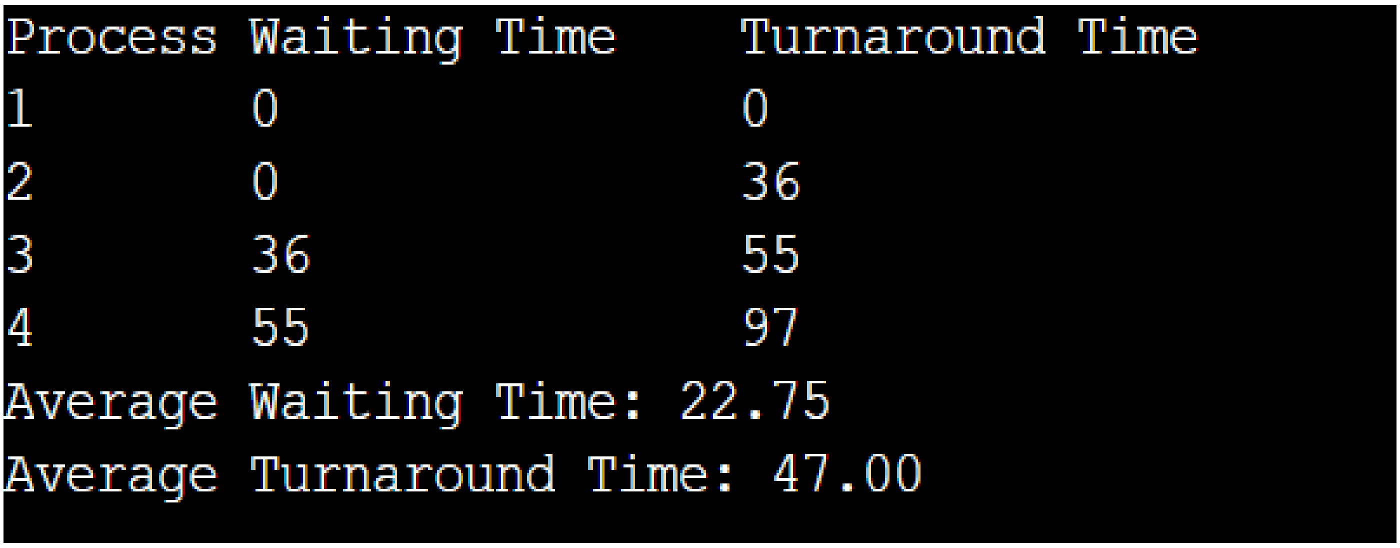


```c
#include <stdio.h>

// Structure to represent a process
struct Process {
    char name;          // Process name (P1, P2, ...)
    int arrivalTime;    // Arrival time
    int burstTime;      // Burst time
    int waitingTime;    // Waiting time
    int turnaroundTime; // Turnaround time
    float priority;     // Priority
};

// Function to calculate waiting time and turnaround time
void calculateWaitingTurnaroundTime(struct Process processes[], int n) {
    int totalWaitingTime = 0;
    int totalTurnaroundTime = 0;

    processes[0].waitingTime = 0; // The first process has no waiting time

    // Calculate waiting time and turnaround time for each process
    for (int i = 1; i < n; i++) {
        processes[i].waitingTime = processes[i - 1].turnaroundTime;
        totalWaitingTime += processes[i].waitingTime;
```



```c
        // Calculate priority for the current process
        processes[i].priority = 1.0 + (float)processes[i].waitingTime / (float)processes[i].burstTime;

        processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;
        totalTurnaroundTime += processes[i].turnaroundTime;
    }

    // Calculate average waiting time and average turnaround time
    float avgWaitingTime = (float)totalWaitingTime / n;
    float avgTurnaroundTime = (float)totalTurnaroundTime / n;

    printf("Process\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%c\t%d\t\t%d\n", processes[i].name, processes[i].waitingTime, processes[i].turnaroundTime);
    }

    printf("Average Waiting Time: %.2f\n", avgWaitingTime);
    printf("Average Turnaround Time: %.2f\n", avgTurnaroundTime);
}

int main() {
    struct Process processes[] = {
        {'P1', 0, 20, 0, 0, 0.0},
        {'P2', 5, 36, 0, 0, 0.0},
        {'P3', 13, 19, 0, 0, 0.0},
        {'P4', 17, 42, 0, 0, 0.0}
    };

    int n = sizeof(processes) / sizeof(processes[0]);

    calculateWaitingTurnaroundTime(processes, n);

    return 0;
}
```

The output of the provided C program. Here's the expected output:



```
Process Waiting Time    Turnaround Time
1       0               0
2       0               36
3       36              55
4       55              97
Average Waiting Time: 22.75
Average Turnaround Time: 47.00
```