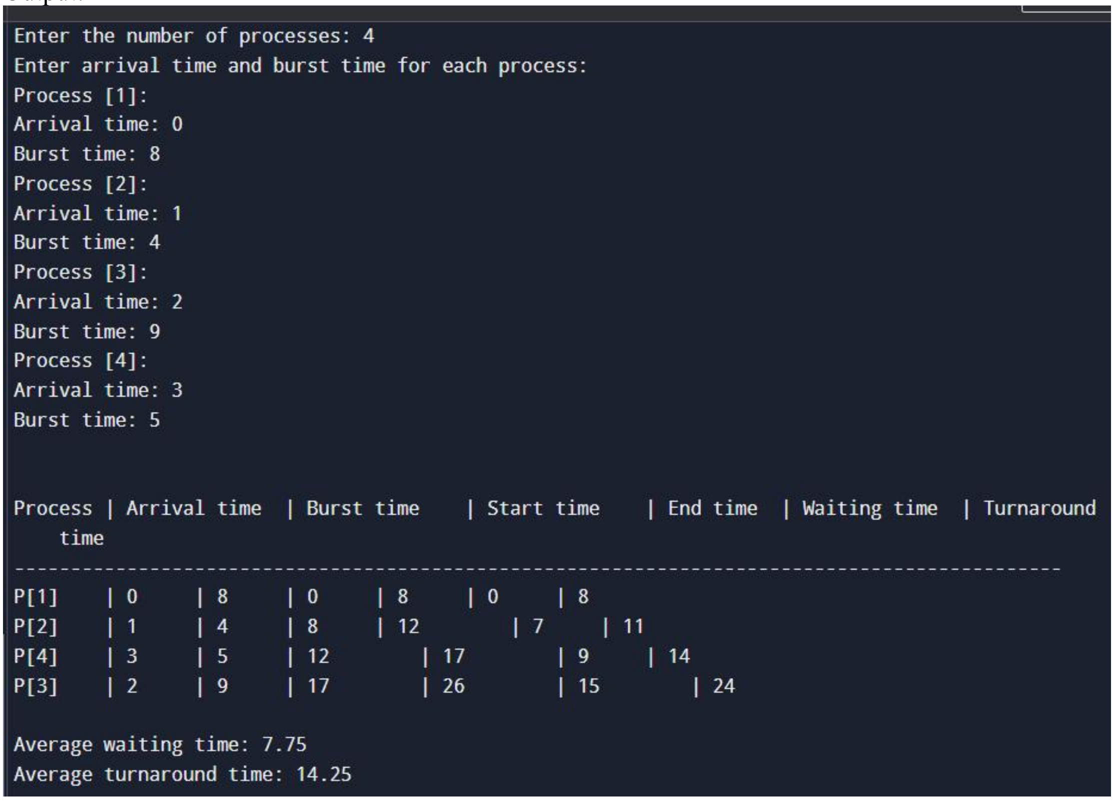
## 7. Construct a C program to implement non-preemptive SJFalgorithm

## Program:

```
#include<stdio.h>

int main() {
    int at[10], bt[10], pr[10];
    int n, i, j, temp, time = 0, count, over = 0;
    int sum_wait = 0, sum_turnaround = 0, start;
    float avgwait, avgturn;

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

    printf("Enter arrival time and burst time for each process:\n");
    for(i = 0; i < n; i++) {
        printf("Process [%d]:\n", i + 1);
        printf("Arrival time: ");
        scanf("%d", &at[i]);
        printf("Burst time: ");
        scanf("%d", &bt[i]);
        pr[i] = i + 1;
    }

    for(i = 0; i < n - 1; i++) {
        for(j = i + 1; j < n; j++) {
            if(at[i] > at[j]) {
                temp = at[i];
                at[i] = at[j];
                at[j] = temp;
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;
                temp = pr[i];
                pr[i] = pr[j];
                pr[j] = temp;
            }
        }
    }

    printf("\n\nProcess\t| Arrival time\t| Burst time\t| Start time\t| End time\t| Waiting time\t| Turnaround time\n");
    printf("------------------------------------------------------------------------------------------\n");

    while(over < n) {
        count = 0;
        for(i = over; i < n; i++) {
            if(at[i] <= time)
                count++;
            else
                break;
        }

        if(count > 1) {
            for(i = over; i < over + count - 1; i++) {
                for(j = i + 1; j < over + count; j++) {
                    if(bt[i] > bt[j]) {
                        temp = at[i];
                        at[i] = at[j];
                        at[j] = temp;
                        temp = bt[i];
                        bt[i] = bt[j];
                        bt[j] = temp;
                        temp = pr[i];
                        pr[i] = pr[j];
                        pr[j] = temp;
                    }
                }
            }
        }

        start = time;
        time += bt[over];
        printf("P[%d]\t| %d\t\t| %d\t\t| %d\t\t| %d\t\t| %d\t\t| %d\n",
               pr[over], at[over], bt[over], start, time,
               time - at[over] - bt[over], time - at[over]);
        sum_wait += time - at[over] - bt[over];
        sum_turnaround += time - at[over];
        over++;
    }

    avgwait = (float)sum_wait / (float)n;
    avgturn = (float)sum_turnaround / (float)n;
    printf("\nAverage waiting time: %.2f\n", avgwait);
    printf("Average turnaround time: %.2f\n", avgturn);

    return 0;
}
```

Output:



```
Enter the number of processes: 4
Enter arrival time and burst time for each process:
Process [1]:
Arrival time: 0
Burst time: 8
Process [2]:
Arrival time: 1
Burst time: 4
Process [3]:
Arrival time: 2
Burst time: 9
Process [4]:
Arrival time: 3
Burst time: 5


Process | Arrival time  | Burst time    | Start time    | End time  | Waiting time  | Turnaround
    time
----------------------------------------------------------------------------------------------
P[1]    | 0     | 8     | 0     | 8     | 0     | 8
P[2]    | 1     | 4     | 8     | 12          | 7     | 11
P[4]    | 3     | 5     | 12          | 17          | 9     | 14
P[3]    | 2     | 9     | 17          | 26          | 15          | 24

Average waiting time: 7.75
Average turnaround time: 14.25
```