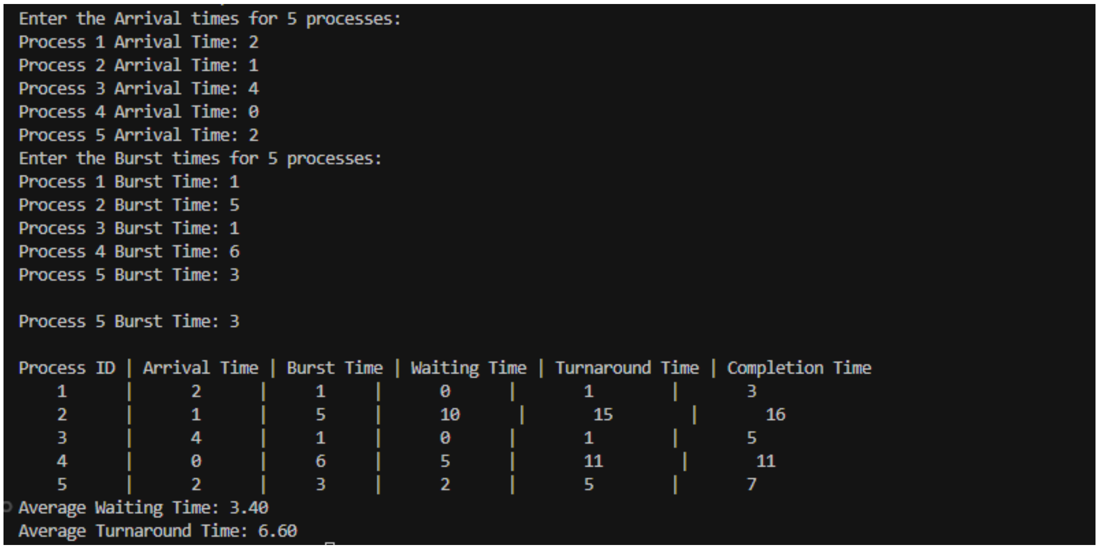
Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.

b) SJF(Pre-emptive and Non - pre-emptive)

Pre-emptive :

```
#include <stdio.h>
int arr[5], bt[5], remaining_bt[5], wt[5], tat[5], pid[5], ct[5];
int totalwt = 0, totaltat = 0, time = 0, completed = 0;
int n = 5;
void main() {
    printf("Enter the Arrival times for 5 processes:\n");
    for (int i = 0; i < n; i++) {
        pid[i] = i + 1;
        printf("Process %d Arrival Time: ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("Enter the Burst times for 5 processes:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d Burst Time: ", i + 1);
        scanf("%d", &bt[i]);
        remaining_bt[i] = bt[i];
    }
    for (int i = 0; i < n; i++) {
        wt[i] = 0;
        tat[i] = 0;
    }
    while (completed != n) {
        int idx = -1;
        int min_bt = 9999;
        for (int i = 0; i < n; i++) {
            if (arr[i] <= time && remaining_bt[i] > 0) {
                if (remaining_bt[i] < min_bt) {
                    min_bt = remaining_bt[i];
                    idx = i;
                } else if (remaining_bt[i] == min_bt) {
                    // If burst time is the same, choose the one with the earlier arrival time
                    if (arr[i] < arr[idx]) {
                        idx = i;
                    }
                }
            }
        }
        if (idx != -1) {
            remaining_bt[idx]--;
            time++;
            if (remaining_bt[idx] == 0) {
                completed++;
                ct[idx] = time;
                tat[idx] = ct[idx] - arr[idx];
                wt[idx] = tat[idx] - bt[idx];
                totalwt += wt[idx];
                totaltat += tat[idx];
            }
        } else {
            time++;
        }
    }
    printf("\nProcess ID | Arrival Time | Burst Time | Waiting Time | Turnaround Time | Completion 
Time\n");
    for (int i = 0; i < n; i++) {
        printf("    %d      |      %d      |     %d     |      %d      |       %d        |       %d\n",
                pid[i], arr[i], bt[i], wt[i], tat[i], ct[i]);
    }
    printf("Average Waiting Time: %.2f\n", (float)totalwt / n);
    printf("Average Turnaround Time: %.2f\n", (float)totaltat / n);
}
```

Output :

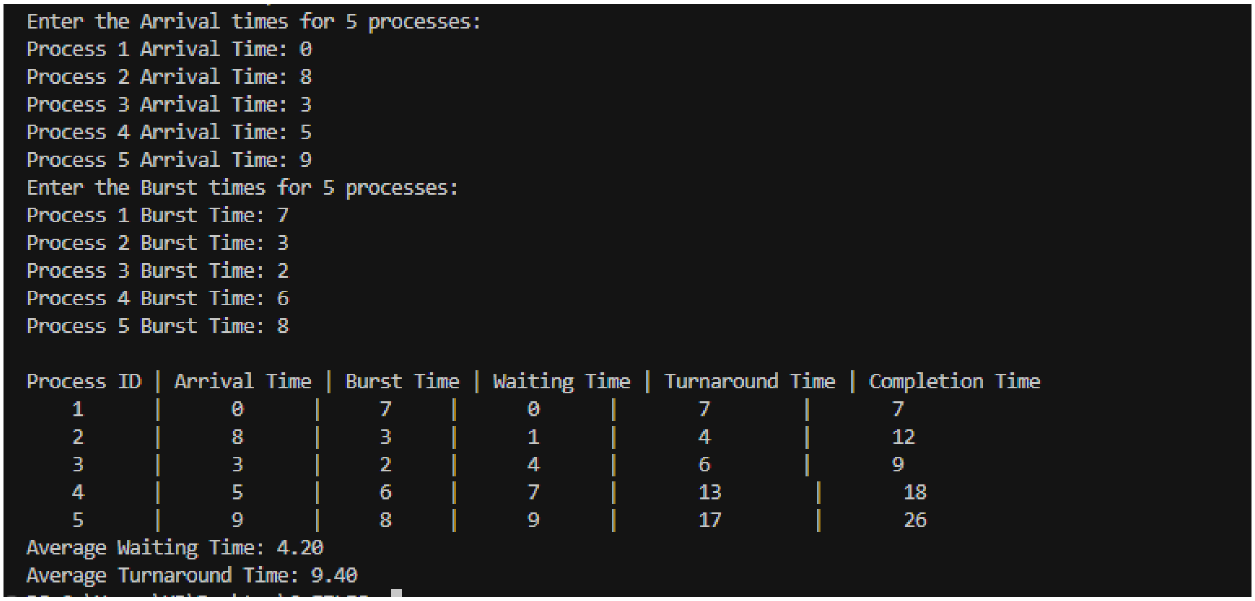


```
Enter the Arrival times for 5 processes:
Process 1 Arrival Time: 2
Process 2 Arrival Time: 1
Process 3 Arrival Time: 4
Process 4 Arrival Time: 0
Process 5 Arrival Time: 2
Enter the Burst times for 5 processes:
Process 1 Burst Time: 1
Process 2 Burst Time: 5
Process 3 Burst Time: 1
Process 4 Burst Time: 6
Process 5 Burst Time: 3

Process 5 Burst Time: 3

Process ID | Arrival Time | Burst Time | Waiting Time | Turnaround Time | Completion Time
    1      |      2      |     1     |      0      |       1        |       3
    2      |      1      |     5     |      10      |       15        |       16
    3      |      4      |     1     |      0      |       1        |       5
    4      |      0      |     6     |      5      |       11        |       11
    5      |      2      |     3     |      2      |       5        |       7
Average Waiting Time: 3.40
Average Turnaround Time: 6.60
```

Non Pre-emptive :

```
#include <stdio.h>

int arr[5], bt[5], wt[5], tat[5], pid[5], ct[5];
int totalwt = 0, totaltat = 0, n = 5;

void main() {
    printf("Enter the Arrival times for 5 processes:\n");
    for (int i = 0; i < n; i++) {
        pid[i] = i + 1;
        printf("Process %d Arrival Time: ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("Enter the Burst times for 5 processes:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d Burst Time: ", i + 1);
        scanf("%d", &bt[i]);
    }

    int completed = 0, time = 0;
    int is_completed[5] = {0};

    while (completed != n) {
        int idx = -1, min_bt = 9999;
        for (int i = 0; i < n; i++) {
            if (arr[i] <= time && !is_completed[i] && bt[i] < min_bt) {
                min_bt = bt[i];
                idx = i;
            }
        }

        if (idx != -1) {
            time += bt[idx];
            ct[idx] = time;
            tat[idx] = ct[idx] - arr[idx];
            wt[idx] = tat[idx] - bt[idx];
            totalwt += wt[idx];
            totaltat += tat[idx];
            is_completed[idx] = 1;
            completed++;
        } else {
            time++;
        }
    }

    printf("\nProcess ID | Arrival Time | Burst Time | Waiting Time | Turnaround Time | Completion Time\n");
    for (int i = 0; i < n; i++) {
        printf("    %d      |      %d      |     %d     |      %d      |       %d        |       %d\n",
               pid[i], arr[i], bt[i], wt[i], tat[i], ct[i]);
    }
    printf("Average Waiting Time: %.2f\n", (float)totalwt / n);
    printf("Average Turnaround Time: %.2f\n", (float)totaltat / n);
}
```

Output :



```
Enter the Arrival times for 5 processes:
Process 1 Arrival Time: 0
Process 2 Arrival Time: 8
Process 3 Arrival Time: 3
Process 4 Arrival Time: 5
Process 5 Arrival Time: 9
Enter the Burst times for 5 processes:
Process 1 Burst Time: 7
Process 2 Burst Time: 3
Process 3 Burst Time: 2
Process 4 Burst Time: 6
Process 5 Burst Time: 8

Process ID | Arrival Time | Burst Time | Waiting Time | Turnaround Time | Completion Time
    1      |      0      |     7     |      0      |       7        |       7
    2      |      8      |     3     |      1      |       4        |       12
    3      |      3      |     2     |      4      |       6        |       9
    4      |      5      |     6     |      7      |       13        |       18
    5      |      9      |     8     |      9      |       17        |       26
Average Waiting Time: 4.20
Average Turnaround Time: 9.40
```