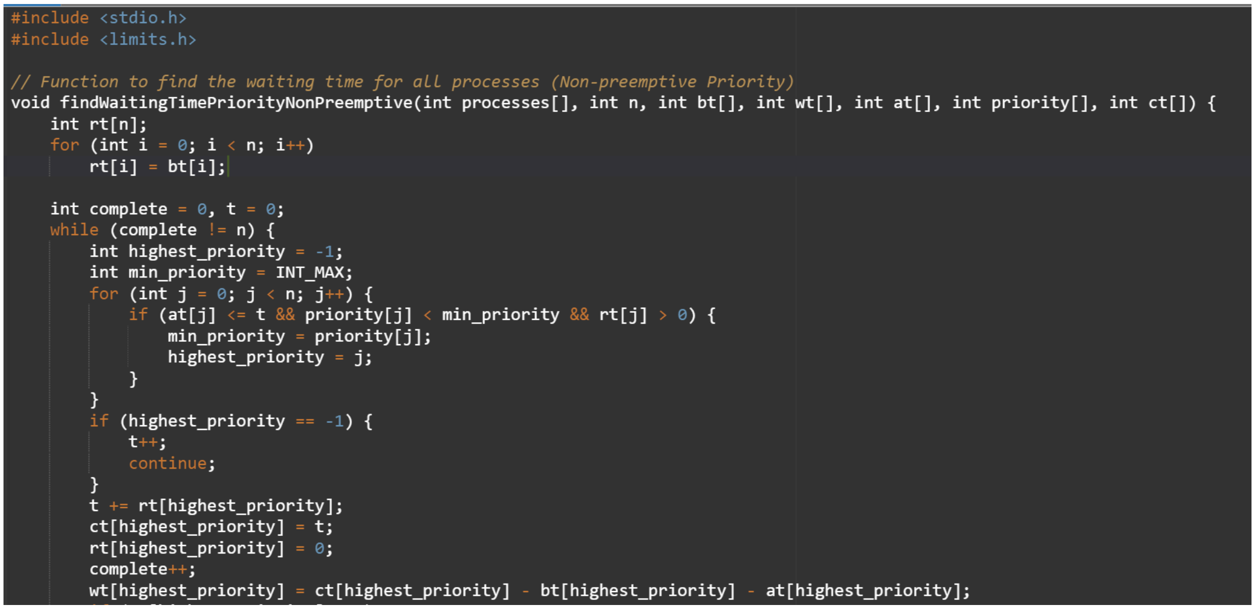
LAB PROGRAM 2

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

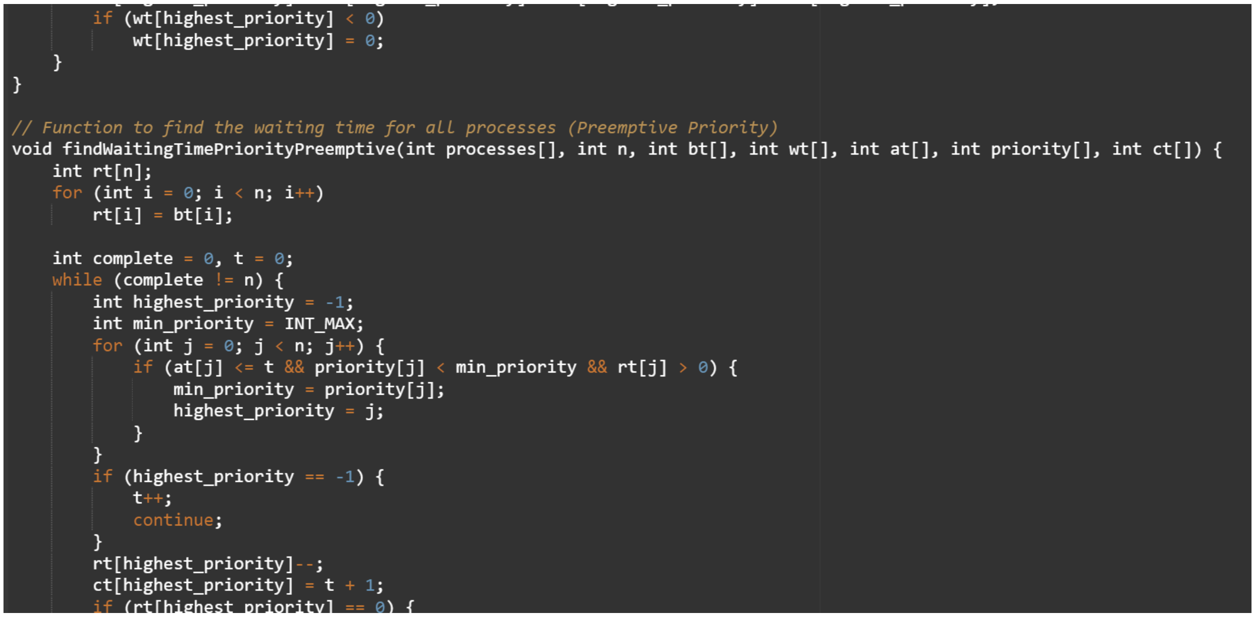
→ Priority (pre-emptive & Non-pre-emptive)

→Round Robin (Experiment with different quantum sizes for RR algorithm)

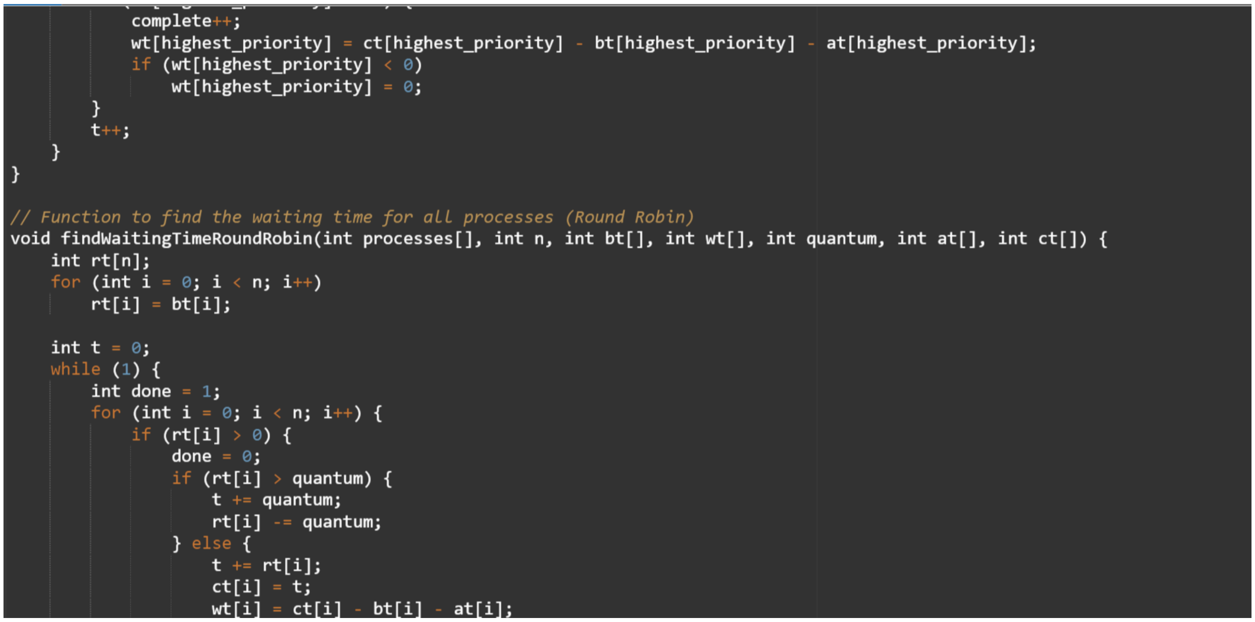
INPUT



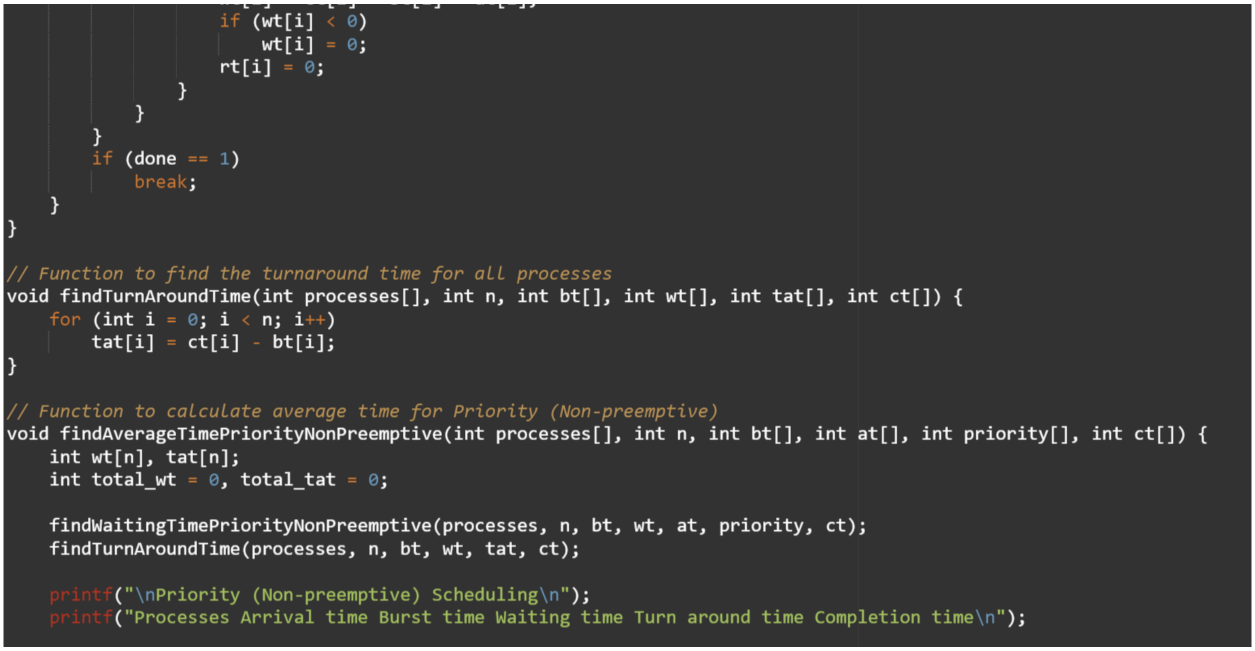
```
#include <stdio.h>
#include <limits.h>

// Function to find the waiting time for all processes (Non-preemptive Priority)
void findWaitingTimePriorityNonPreemptive(int processes[], int n, int bt[], int wt[], int at[], int priority[], int ct[]) {
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = bt[i];

    int complete = 0, t = 0;
    while (complete != n) {
        int highest_priority = -1;
        int min_priority = INT_MAX;
        for (int j = 0; j < n; j++) {
            if (at[j] <= t && priority[j] < min_priority && rt[j] > 0) {
                min_priority = priority[j];
                highest_priority = j;
            }
        }
        if (highest_priority == -1) {
            t++;
            continue;
        }
        t += rt[highest_priority];
        ct[highest_priority] = t;
        rt[highest_priority] = 0;
        complete++;
        wt[highest_priority] = ct[highest_priority] - bt[highest_priority] - at[highest_priority];
```



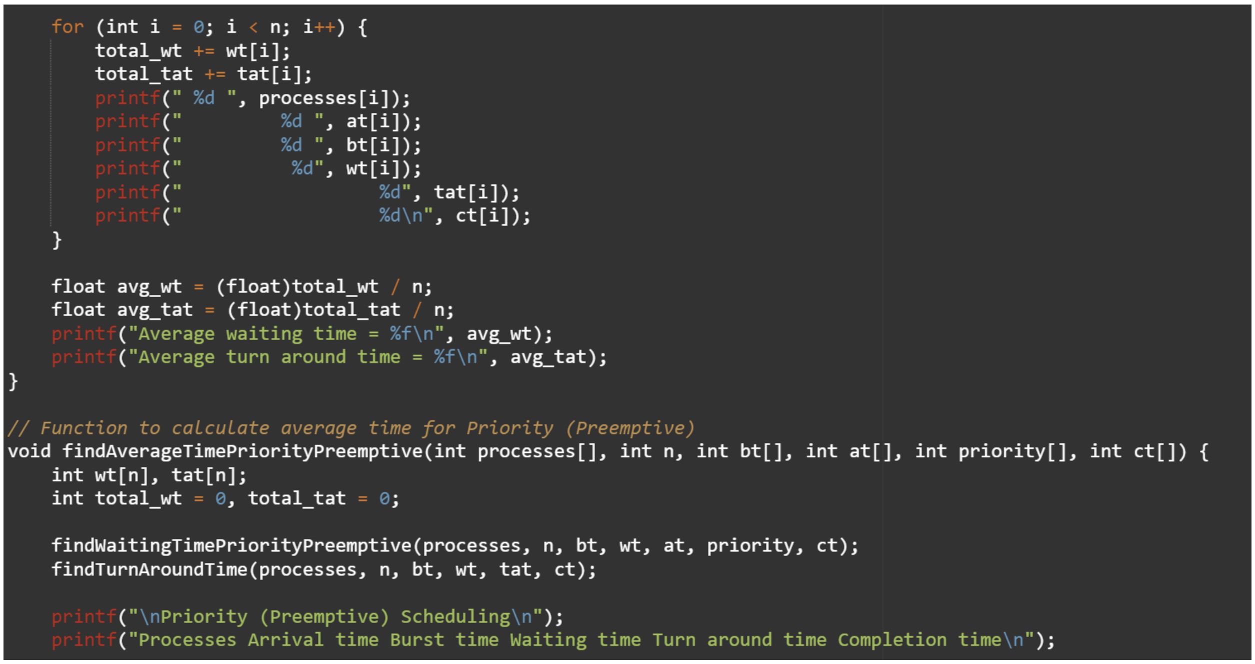
```
        if (wt[highest_priority] < 0)
            wt[highest_priority] = 0;
    }
}

// Function to find the waiting time for all processes (Preemptive Priority)
void findWaitingTimePriorityPreemptive(int processes[], int n, int bt[], int wt[], int at[], int priority[], int ct[]) {
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = bt[i];

    int complete = 0, t = 0;
    while (complete != n) {
        int highest_priority = -1;
        int min_priority = INT_MAX;
        for (int j = 0; j < n; j++) {
            if (at[j] <= t && priority[j] < min_priority && rt[j] > 0) {
                min_priority = priority[j];
                highest_priority = j;
            }
        }
        if (highest_priority == -1) {
            t++;
            continue;
        }
        rt[highest_priority]--;
        ct[highest_priority] = t + 1;
        if (rt[highest_priority] == 0) {
```



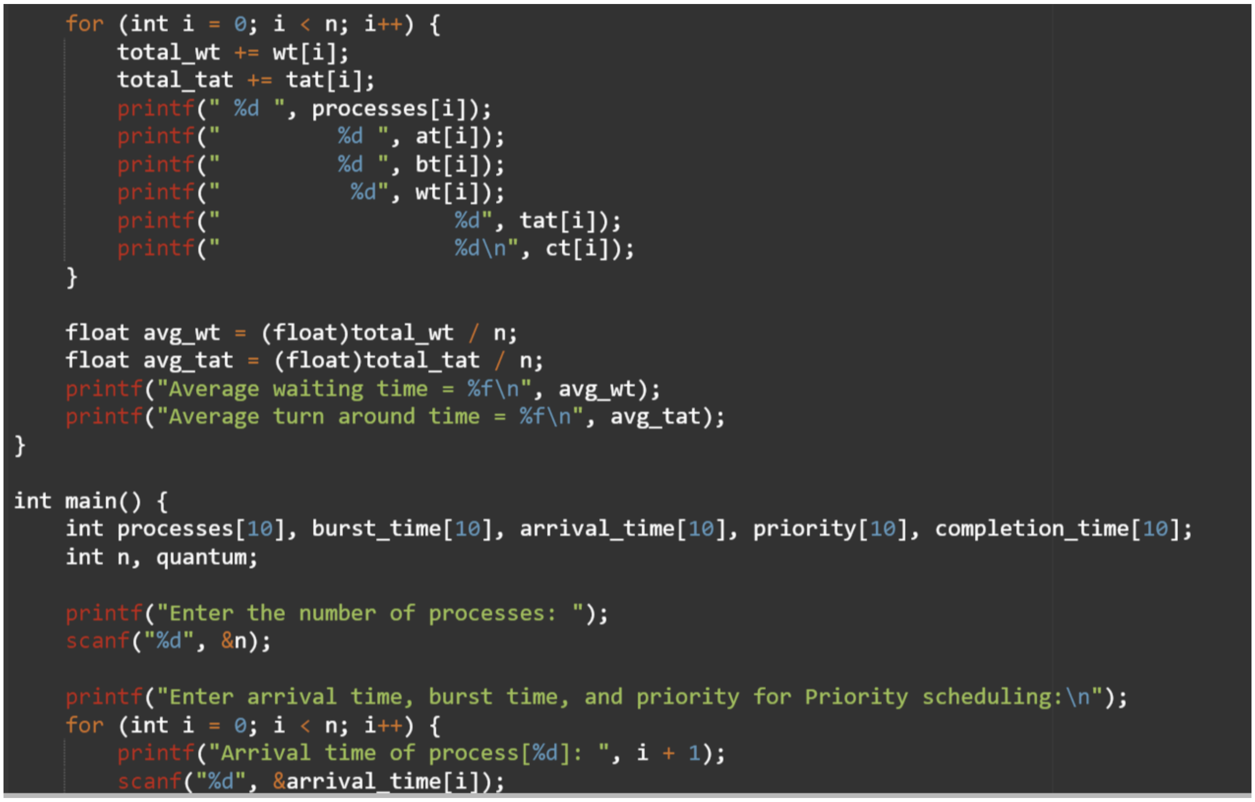
```
            complete++;
            wt[highest_priority] = ct[highest_priority] - bt[highest_priority] - at[highest_priority];
            if (wt[highest_priority] < 0)
                wt[highest_priority] = 0;
        }
        t++;
    }
}

// Function to find the waiting time for all processes (Round Robin)
void findWaitingTimeRoundRobin(int processes[], int n, int bt[], int wt[], int quantum, int at[], int ct[]) {
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = bt[i];

    int t = 0;
    while (1) {
        int done = 1;
        for (int i = 0; i < n; i++) {
            if (rt[i] > 0) {
                done = 0;
                if (rt[i] > quantum) {
                    t += quantum;
                    rt[i] -= quantum;
                } else {
                    t += rt[i];
                    ct[i] = t;
                    wt[i] = ct[i] - bt[i] - at[i];
```



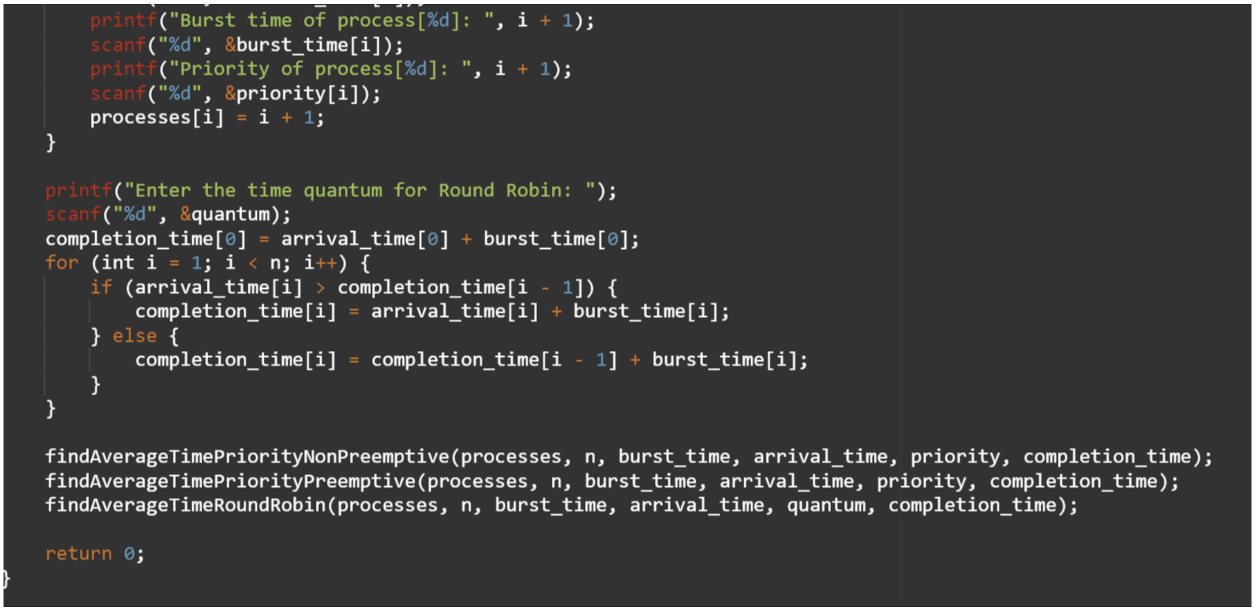
```
                    if (wt[i] < 0)
                        wt[i] = 0;
                    rt[i] = 0;
                }
            }
        }
        if (done == 1)
            break;
    }
}

// Function to find the turnaround time for all processes
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[], int ct[]) {
    for (int i = 0; i < n; i++)
        tat[i] = ct[i] - bt[i];
}

// Function to calculate average time for Priority (Non-preemptive)
void findAverageTimePriorityNonPreemptive(int processes[], int n, int bt[], int at[], int priority[], int ct[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;

    findWaitingTimePriorityNonPreemptive(processes, n, bt, wt, at, priority, ct);
    findTurnAroundTime(processes, n, bt, wt, tat, ct);

    printf("\nPriority (Non-preemptive) Scheduling\n");
    printf("Processes Arrival time Burst time Waiting time Turn around time Completion time\n");
```



```
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf(" %d ", processes[i]);
        printf("         %d ", at[i]);
        printf("         %d ", bt[i]);
        printf("          %d", wt[i]);
        printf("                  %d", tat[i]);
        printf("                  %d\n", ct[i]);
    }

    float avg_wt = (float)total_wt / n;
    float avg_tat = (float)total_tat / n;
    printf("Average waiting time = %f\n", avg_wt);
    printf("Average turn around time = %f\n", avg_tat);
}

// Function to calculate average time for Priority (Preemptive)
void findAverageTimePriorityPreemptive(int processes[], int n, int bt[], int at[], int priority[], int ct[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;

    findWaitingTimePriorityPreemptive(processes, n, bt, wt, at, priority, ct);
    findTurnAroundTime(processes, n, bt, wt, tat, ct);

    printf("\nPriority (Preemptive) Scheduling\n");
    printf("Processes Arrival time Burst time Waiting time Turn around time Completion time\n");
```

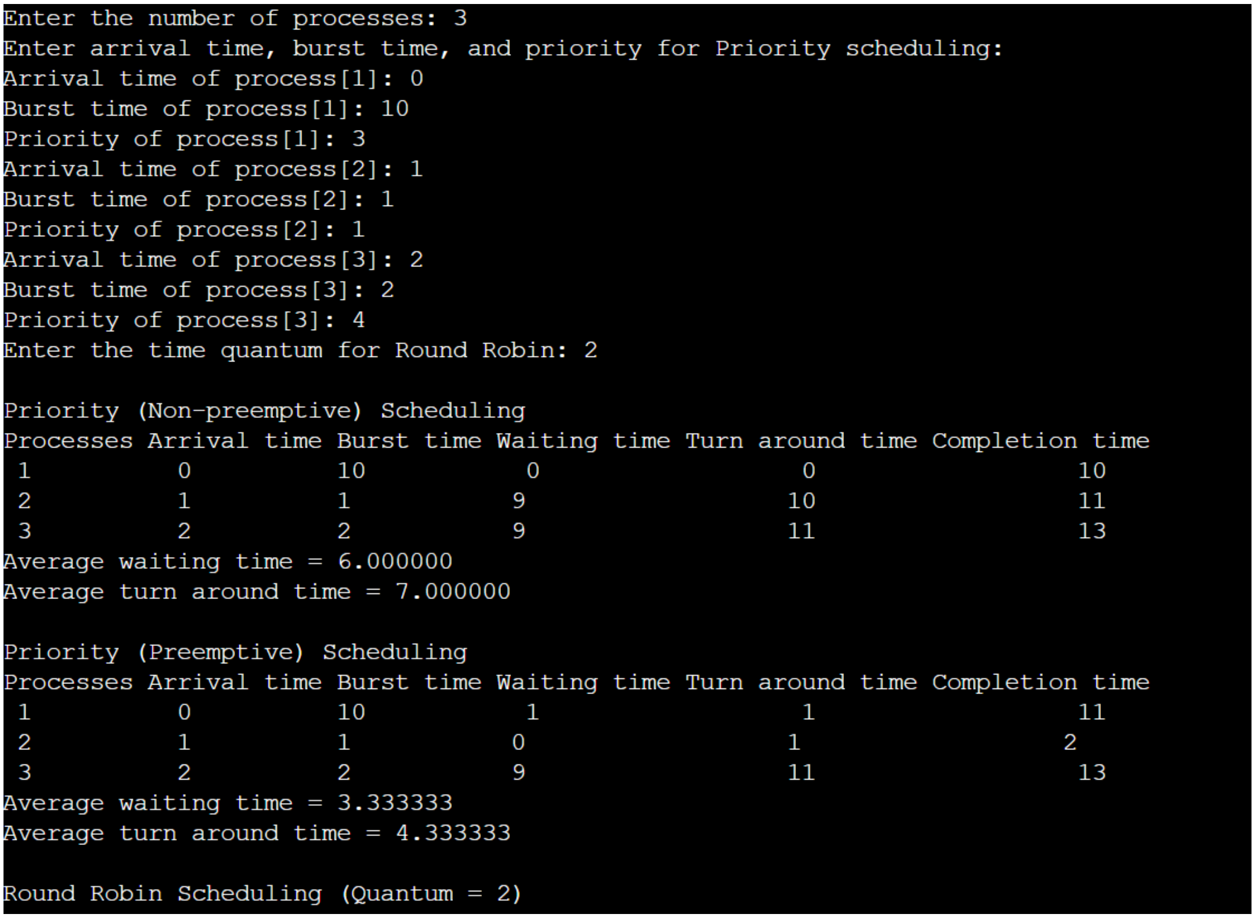


```c
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf(" %d ", processes[i]);
        printf("         %d ", at[i]);
        printf("         %d ", bt[i]);
        printf("          %d", wt[i]);
        printf("                  %d", tat[i]);
        printf("                  %d\n", ct[i]);
    }

    float avg_wt = (float)total_wt / n;
    float avg_tat = (float)total_tat / n;
    printf("Average waiting time = %f\n", avg_wt);
    printf("Average turn around time = %f\n", avg_tat);
}

int main() {
    int processes[10], burst_time[10], arrival_time[10], priority[10], completion_time[10];
    int n, quantum;

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

    printf("Enter arrival time, burst time, and priority for Priority scheduling:\n");
    for (int i = 0; i < n; i++) {
        printf("Arrival time of process[%d]: ", i + 1);
        scanf("%d", &arrival_time[i]);
```

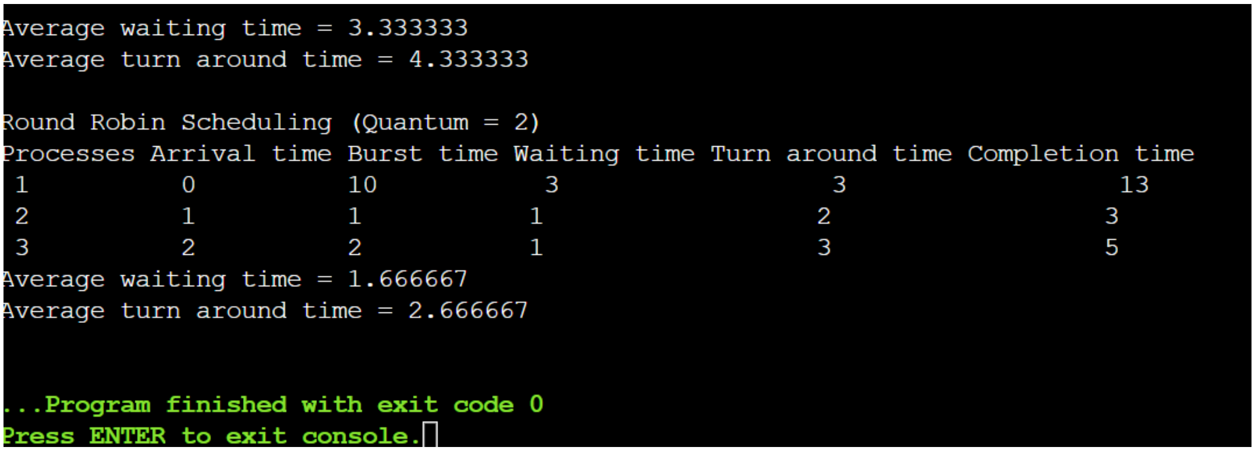


```c
        printf("Burst time of process[%d]: ", i + 1);
        scanf("%d", &burst_time[i]);
        printf("Priority of process[%d]: ", i + 1);
        scanf("%d", &priority[i]);
        processes[i] = i + 1;
    }

    printf("Enter the time quantum for Round Robin: ");
    scanf("%d", &quantum);
    completion_time[0] = arrival_time[0] + burst_time[0];
    for (int i = 1; i < n; i++) {
        if (arrival_time[i] > completion_time[i - 1]) {
            completion_time[i] = arrival_time[i] + burst_time[i];
        } else {
            completion_time[i] = completion_time[i - 1] + burst_time[i];
        }
    }

    findAverageTimePriorityNonPreemptive(processes, n, burst_time, arrival_time, priority, completion_time);
    findAverageTimePriorityPreemptive(processes, n, burst_time, arrival_time, priority, completion_time);
    findAverageTimeRoundRobin(processes, n, burst_time, arrival_time, quantum, completion_time);

    return 0;
}
```

OUTPUT



```
Enter the number of processes: 3
Enter arrival time, burst time, and priority for Priority scheduling:
Arrival time of process[1]: 0
Burst time of process[1]: 10
Priority of process[1]: 3
Arrival time of process[2]: 1
Burst time of process[2]: 1
Priority of process[2]: 1
Arrival time of process[3]: 2
Burst time of process[3]: 2
Priority of process[3]: 4
Enter the time quantum for Round Robin: 2

Priority (Non-preemptive) Scheduling
Processes Arrival time Burst time Waiting time Turn around time Completion time
 1          0          10           0                  0                  10
 2          1          1           9                  10                  11
 3          2          2           9                  11                  13
Average waiting time = 6.000000
Average turn around time = 7.000000

Priority (Preemptive) Scheduling
Processes Arrival time Burst time Waiting time Turn around time Completion time
 1          0          10           1                  1                  11
 2          1          1           0                  1                  2
 3          2          2           9                  11                  13
Average waiting time = 3.333333
Average turn around time = 4.333333

Round Robin Scheduling (Quantum = 2)
```



```
Average waiting time = 3.333333
Average turn around time = 4.333333

Round Robin Scheduling (Quantum = 2)
Processes Arrival time Burst time Waiting time Turn around time Completion time
 1          0          10           3                  3                  13
 2          1          1           1                  2                  3
 3          2          2           1                  3                  5
Average waiting time = 1.666667
Average turn around time = 2.666667


...Program finished with exit code 0
Press ENTER to exit console.
```