```c
#include <stdio.h>

struct Process {
    int pid;      // Process ID
    int arrival;  // Arrival time
    int burst;    // Burst time
    int remaining; // Remaining time
    int waiting;  // Waiting time
    int turnaround;// Turnaround time
};

int main() {
    int n, i, time = 0, tq;
    struct Process p[10];
    int completed = 0;

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

    for (i = 0; i < n; i++) {
        p[i].pid = i + 1;
        printf("Enter Arrival Time of P%d: ", i + 1);
        scanf("%d", &p[i].arrival);
        printf("Enter Burst Time of P%d: ", i + 1);
        scanf("%d", &p[i].burst);
        p[i].remaining = p[i].burst;
        p[i].waiting = 0;
        p[i].turnaround = 0;
    }

    printf("Enter Time Quantum: ");
```

```
scanf("%d", &tq);

int queue[20], front = 0, rear = 0;
int visited[10] = {0};

// Add first arrived process
int min_arrival = 9999, start_index = 0;
for (i = 0; i < n; i++) {
   if (p[i].arrival < min_arrival) {
      min_arrival = p[i].arrival;
      start_index = i;
   }
}
time = p[start_index].arrival;
queue[rear++] = start_index;
visited[start_index] = 1;

printf("\nGantt Chart:\n");

while (completed != n) {
   int idx = queue[front++];
   printf("| P%d (%d to ", p[idx].pid, time);

   if (p[idx].remaining <= tq) {
      time += p[idx].remaining;
      p[idx].remaining = 0;
      completed++;

      printf("%d) ", time);

      p[idx].turnaround = time - p[idx].arrival;
```

```
            p[idx].waiting = p[idx].turnaround - p[idx].burst;
        } else {
            p[idx].remaining -= tq;
            time += tq;
            printf("%d) ", time);
        }

        // Add newly arrived processes
        for (i = 0; i < n; i++) {
            if (p[i].arrival <= time && visited[i] == 0 && p[i].remaining > 0) {
                queue[rear++] = i;
                visited[i] = 1;
            }
        }

        // Requeue the current process if not finished
        if (p[idx].remaining > 0)
            queue[rear++] = idx;

        // If queue is empty, jump to next process arrival
        if (front == rear) {
            for (i = 0; i < n; i++) {
                if (p[i].remaining > 0) {
                    queue[rear++] = i;
                    visited[i] = 1;
                    time = p[i].arrival;
                    break;
                }
            }
        }
    }
```

```
    printf("|\n");

    float total_wait = 0, total_turnaround = 0;

    printf("\n%-10s%-15s%-15s%-15s%-15s\n", "Process", "Arrival", "Burst", "Waiting",
"Turnaround");
    for (i = 0; i < n; i++) {
        printf("P%-9d%-15d%-15d%-15d%-15d\n", p[i].pid, p[i].arrival, p[i].burst, p[i].waiting,
p[i].turnaround);
        total_wait += p[i].waiting;
        total_turnaround += p[i].turnaround;
    }

    printf("\nAverage Waiting Time: %.2f", total_wait / n);
    printf("\nAverage Turnaround Time: %.2f\n", total_turnaround / n);

    return 0;
}
```

OUTPUT

```
Enter number of processes: 3
Enter Arrival Time of P1: 0
Enter Burst Time of P1: 5
Enter Arrival Time of P2: 1
Enter Burst Time of P2: 4
Enter Arrival Time of P3: 2
Enter Burst Time of P3: 2
Enter Time Quantum: 2

Gantt Chart:
| P1 (0 to 2) | P2 (2 to 4) | P3 (4 to 6) | P1 (6 to 8) | P2 (8 to 10) | P1 (10 to 11) |
```

Process	Arrival	Burst	Waiting	Turnaround
P1	0	5	6	11
P2	1	4	5	9
P3	2	2	2	4

Average Waiting Time: 4.33

Average Turnaround Time: 8.00