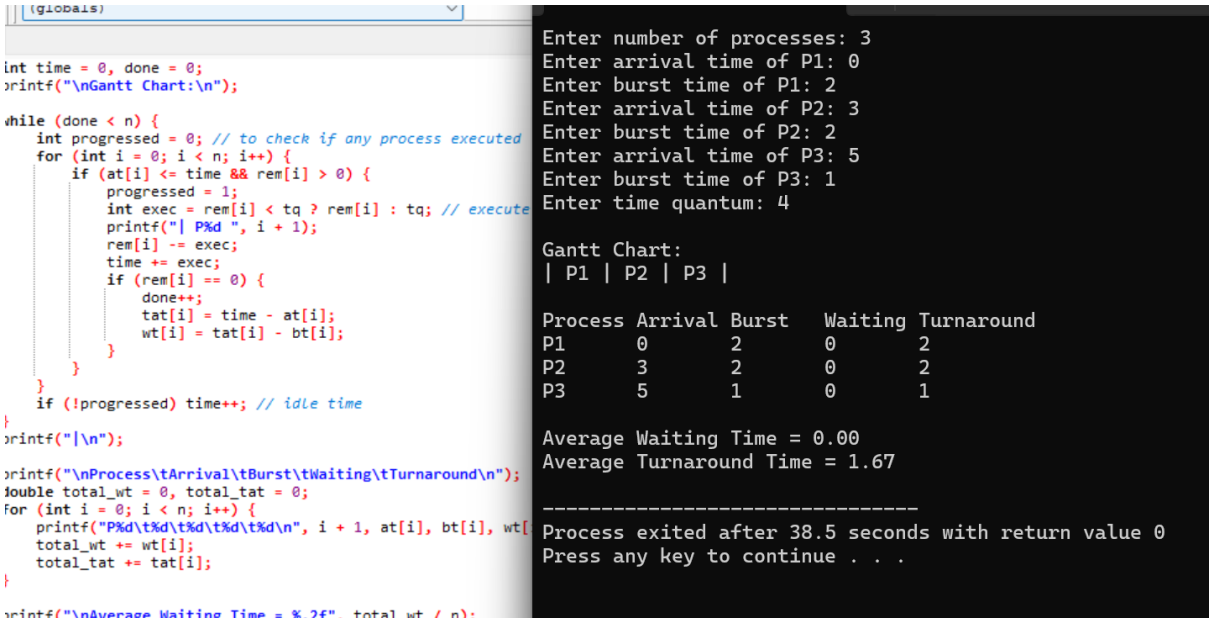
8. Construct a C program to simulate Round Robin scheduling algorithm with C.

```
#include <stdio.h>

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

   int at[n], bt[n], rem[n], wt[n], tat[n];
   for (int i = 0; i < n; i++) {
      printf("Enter arrival time of P%d: ", i + 1);
      scanf("%d", &at[i]);
      printf("Enter burst time of P%d: ", i + 1);
      scanf("%d", &bt[i]);
      rem[i] = bt[i]; // remaining burst time
      wt[i] = 0;
      tat[i] = 0;
   }

   printf("Enter time quantum: ");
   scanf("%d", &tq);

   int time = 0, done = 0;
   printf("\nGantt Chart:\n");

   while (done < n) {
      int progressed = 0; // to check if any process executed
      for (int i = 0; i < n; i++) {
         if (at[i] <= time && rem[i] > 0) {
            progressed = 1;
            int exec = rem[i] < tq ? rem[i] : tq; // execute min(rem, tq)
            printf("| P%d ", i + 1);
            rem[i] -= exec;
            time += exec;
            if (rem[i] == 0) {
               done++;
               tat[i] = time - at[i];
               wt[i] = tat[i] - bt[i];
            }
         }
      }
      if (!progressed) time++; // idle time
   }
   printf("|\n");
```

```
    printf("\nProcess\tArrival\tBurst\tWaiting\tTurnaround\n");
    double total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\n", i + 1, at[i], bt[i], wt[i], tat[i]);
        total_wt += wt[i];
        total_tat += tat[i];
    }

    printf("\nAverage Waiting Time = %.2f", total_wt / n);
    printf("\nAverage Turnaround Time = %.2f\n", total_tat / n);

    return 0;
}
```

OUTPUT:



```
int time = 0, done = 0;
printf("\nGantt Chart:\n");

while (done < n) {
    int progressed = 0; // to check if any process executed
    for (int i = 0; i < n; i++) {
        if (at[i] <= time && rem[i] > 0) {
            progressed = 1;
            int exec = rem[i] < tq ? rem[i] : tq; // execute
            printf("| P%d ", i + 1);
            rem[i] -= exec;
            time += exec;
            if (rem[i] == 0) {
                done++;
                tat[i] = time - at[i];
                wt[i] = tat[i] - bt[i];
            }
        }
    }
    if (!progressed) time++; // idle time
}
printf("|\n");

printf("\nProcess\tArrival\tBurst\tWaiting\tTurnaround\n");
double total_wt = 0, total_tat = 0;
for (int i = 0; i < n; i++) {
    printf("P%d\t%d\t%d\t%d\t%d\n", i + 1, at[i], bt[i], wt[
    total_wt += wt[i];
    total_tat += tat[i];
}

printf("\nAverage Waiting Time = %.2f", total_wt / n);
```

```
Enter number of processes: 3
Enter arrival time of P1: 0
Enter burst time of P1: 2
Enter arrival time of P2: 3
Enter burst time of P2: 2
Enter arrival time of P3: 5
Enter burst time of P3: 1
Enter time quantum: 4

Gantt Chart:
| P1 | P2 | P3 |

Process Arrival Burst   Waiting Turnaround
P1      0       2       0       2
P2      3       2       0       2
P3      5       1       0       1

Average Waiting Time = 0.00
Average Turnaround Time = 1.67

--------------------------------
Process exited after 38.5 seconds with return value 0
Press any key to continue . . .
```