7. Construct a C program to implement non-preemptive SJF algorithm.

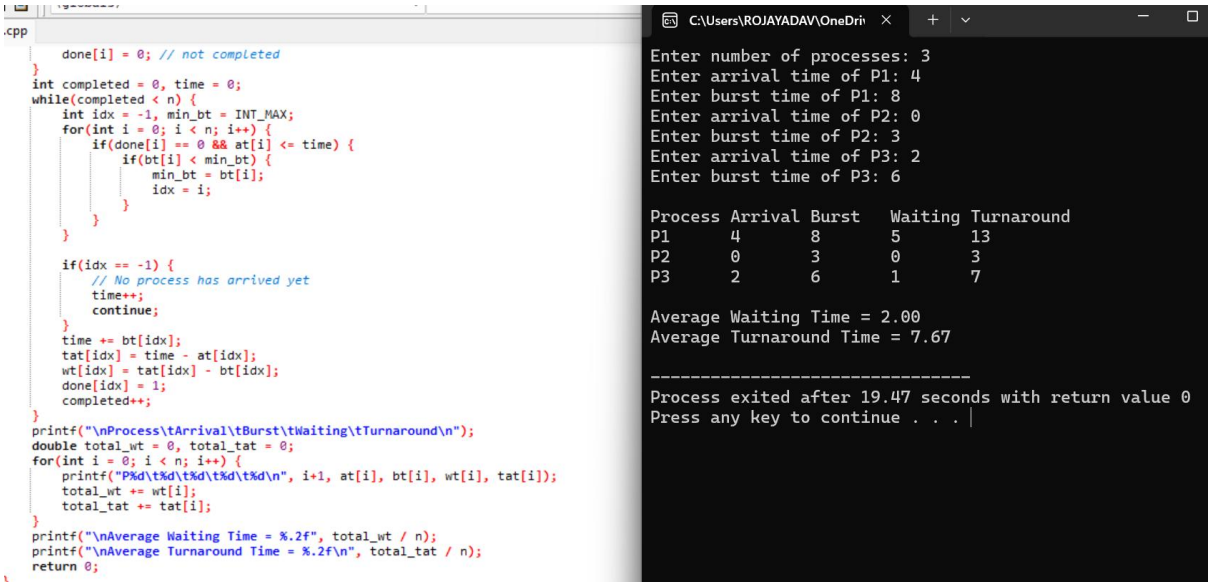
```
#include <stdio.h>
#include <limits.h>
int main() {
   int n;
   printf("Enter number of processes: ");
   scanf("%d", &n);
   int at[n], bt[n], done[n];
   int wt[n], tat[n];
   for(int i = 0; i < n; i++) {
      printf("Enter arrival time of P%d: ", i + 1);
      scanf("%d", &at[i]);
      printf("Enter burst time of P%d: ", i + 1);
      scanf("%d", &bt[i]);
      done[i] = 0; // not completed
   }
   int completed = 0, time = 0;
   while(completed < n) {
      int idx = -1, min_bt = INT_MAX;
      for(int i = 0; i < n; i++) {
         if(done[i] == 0 && at[i] <= time) {
            if(bt[i] < min_bt) {
               min_bt = bt[i];
               idx = i;
            }
         }
      }

      if(idx == -1) {
         // No process has arrived yet
         time++;
         continue;
      }
      time += bt[idx];
      tat[idx] = time - at[idx];
      wt[idx] = tat[idx] - bt[idx];
      done[idx] = 1;
      completed++;
   }
   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;
}
```

INPUT:

Enter number of processes: 3
Enter arrival time of P1: 0
Enter burst time of P1: 10
Enter arrival time of P2: 1
Enter burst time of P2: 5
Enter arrival time of P3: 2
Enter burst time of P3: 8

OUTPUT:



```
.cpp
    done[i] = 0; // not completed
}
int completed = 0, time = 0;
while(completed < n) {
    int idx = -1, min_bt = INT_MAX;
    for(int i = 0; i < n; i++) {
        if(done[i] == 0 && at[i] <= time) {
            if(bt[i] < min_bt) {
                min_bt = bt[i];
                idx = i;
            }
        }
    }

    if(idx == -1) {
        // No process has arrived yet
        time++;
        continue;
    }
    time += bt[idx];
    tat[idx] = time - at[idx];
    wt[idx] = tat[idx] - bt[idx];
    done[idx] = 1;
    completed++;
}
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;
```

```
C:\Users\ROJAYADAV\OneDri
Enter number of processes: 3
Enter arrival time of P1: 4
Enter burst time of P1: 8
Enter arrival time of P2: 0
Enter burst time of P2: 3
Enter arrival time of P3: 2
Enter burst time of P3: 6

Process Arrival Burst   Waiting Turnaround
P1      4       8       5       13
P2      0       3       0       3
P3      2       6       1       7

Average Waiting Time = 2.00
Average Turnaround Time = 7.67

--------------------------------
Process exited after 19.47 seconds with return value 0
Press any key to continue . . .
```