```c
// C program to implement Shortest Remaining Time First
// Shortest Remaining Time First (SRTF)

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

struct Process {
    int pid; // Process ID
    int bt; // Burst Time
    int art; // Arrival Time
};

// Function to find the waiting time for all
// processes
void findWaitingTime(struct Process proc[], int n, int wt[]) {
    int rt[n];

    // Copy the burst time into rt[]
    for (int i = 0; i < n; i++)
        rt[i] = proc[i].bt;

    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    int check = 0; // changed boolean to integer

    // Process until all processes gets
    // completed
    while (complete != n) {

        // Find process with minimum
        // remaining time among the
        // processes that arrives till the
        // current time
        for (int j = 0; j < n; j++) {
            if ((proc[j].art <= t) &&
            (rt[j] < minm) && rt[j] > 0) {
                minm = rt[j];
                shortest = j;
                check = 1; // changed boolean to integer
            }
        }

        if (check == 0) {
            t++;
            continue;
        }

        // Reduce remaining time by one
        rt[shortest]--;

        // Update minimum
        minm = rt[shortest];
```

```
        if (minm == 0)
            minm = INT_MAX;

        // If a process gets completely
        // executed
        if (rt[shortest] == 0) {

            // Increment complete
            complete++;
            check = 0; // changed boolean to integer

            // Find finish time of current
            // process
            finish_time = t + 1;

            // Calculate waiting time
            wt[shortest] = finish_time -
                        proc[shortest].bt -
                        proc[shortest].art;

            if (wt[shortest] < 0)
                wt[shortest] = 0;
        }
        // Increment time
        t++;
    }
}

// Function to calculate turn around time
void findTurnAroundTime(struct Process proc[], int n, int wt[], int tat[]) {
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n; i++)
        tat[i] = proc[i].bt + wt[i];
}

// Function to calculate average time
void findavgTime(struct Process proc[], int n) {
    int wt[n], tat[n], total_wt = 0,
                    total_tat = 0;

    // Function to find waiting time of all
    // processes
    findWaitingTime(proc, n, wt);

    // Function to find turn around time for
    // all processes
    findTurnAroundTime(proc, n, wt, tat);

    // Display processes along with all
    // details
    printf(" P\t\t"
```

```
        "BT\t\t"
        "WT\t\t"
        "TAT\t\t\n");

    // Calculate total waiting time and
    // total turnaround time
    for (int i = 0; i < n; i++) {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf(" %d\t\t"
            "%d\t\t %d"
            "\t\t %d\n", proc[i].pid,
            proc[i].bt, wt[i], tat[i]);
    }

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

// Driver code
int main() {
    struct Process proc[] = { { 1, 6, 2 }, { 2, 2, 5 },
                    { 3, 8, 1 }, { 4, 3, 0}, {5, 4, 4} };
    int n = sizeof(proc) / sizeof(proc[0]);

    findavgTime(proc, n);
    return 0;
}
```