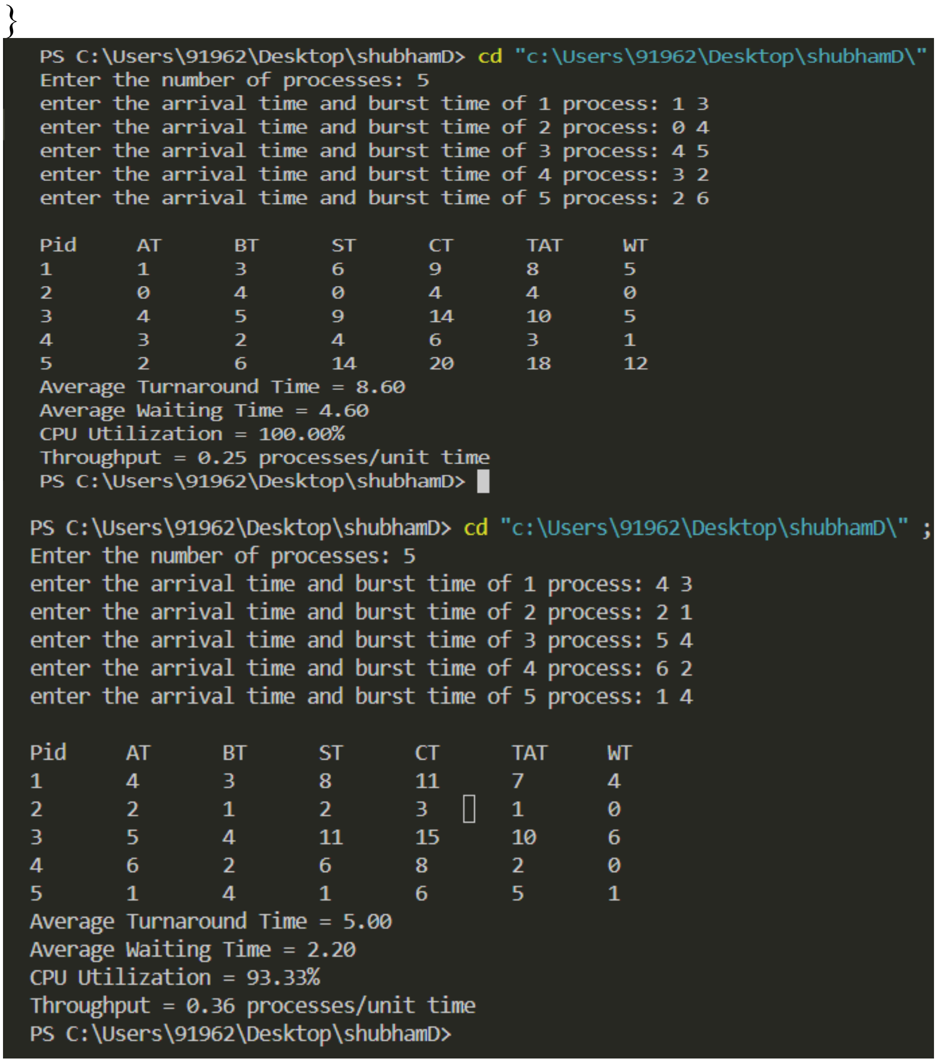
6- C program to implement SRTF cpu scheduling algorithm.

Shortest Remaining Time First (SRTF) is a CPU scheduling algorithm used in operating systems to efficiently schedule processes. It is also known as **Preemptive Shortest Job Next (SJN)** It is a preemptive algorithm that selects the process with the smallest remaining burst time to execute next.

code-

```
#include <stdio.h>

struct process {
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
};

int main() {
    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    float cpu_utilisation;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_idle_time = 0;
    float throughput;
    int burst_remaining[100];
    int is_completed[100];

    for (int i = 0; i < 100; i++) {
        is_completed[i] = 0;
    }

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

    for (int i = 0; i < n; i++) {
        printf("enter the arrival time and burst time of %d process: ",i+1);
        scanf("%d %d",  &p[i].arrival_time,&p[i].burst_time);
        p[i].pid = i + 1;
        burst_remaining[i] = p[i].burst_time;
    }

    int current_time = 0;
    int completed = 0;
    int prev = 0;
    while (completed != n) {
        int idx = -1;
        int mn = 10000000;

        for (int i = 0; i < n; i++) {
            if (p[i].arrival_time <= current_time && is_completed[i] == 0) {
                if (burst_remaining[i] < mn) {
                    mn = burst_remaining[i];
                    idx = i;
                }
                if (burst_remaining[i] == mn) {
                    if (p[i].arrival_time < p[idx].arrival_time) {
                        mn = burst_remaining[i];
                        idx = i;
                    }
                }
            }
        }

        if (idx != -1) {
            if (burst_remaining[idx] == p[idx].burst_time) {
                p[idx].start_time = current_time;
                total_idle_time += p[idx].start_time - prev;
            }
            burst_remaining[idx] -= 1;
            current_time++;
            prev = current_time;

            if (burst_remaining[idx] == 0) {
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time -
p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;

                total_turnaround_time += p[idx].turnaround_time;
                total_waiting_time += p[idx].waiting_time;

                is_completed[idx] = 1;
                completed++;
            }
        } else {
            current_time++;
        }
    }
    int min_arrival_time = 10000000;
    int max_completion_time = -1;

    for (int i = 0; i < n; i++) {
        min_arrival_time = (p[i].arrival_time < min_arrival_time) ?
p[i].arrival_time : min_arrival_time;
        max_completion_time = (p[i].completion_time > max_completion_time) ?
p[i].completion_time : max_completion_time;
    }
    avg_turnaround_time = (float)total_turnaround_time / n;
    avg_waiting_time = (float)total_waiting_time / n;
    cpu_utilisation = ((float)(max_completion_time - total_idle_time) /
max_completion_time) * 100;
    throughput = (float)n / (max_completion_time - min_arrival_time);

    printf("\nPid\tAT\tBT\tST\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].arrival_time,
p[i].burst_time, p[i].start_time, p[i].completion_time, p[i].turnaround_time,
p[i].waiting_time);

    printf("Average Turnaround Time = %.2f\n", avg_turnaround_time);
    printf("Average Waiting Time = %.2f\n", avg_waiting_time);
    printf("CPU Utilization = %.2f%%\n", cpu_utilisation);
    printf("Throughput = %.2f processes/unit time\n", throughput);

    return 0;
}
```



```
PS C:\Users\91962\Desktop\shubhamD> cd "c:\Users\91962\Desktop\shubhamD\"
Enter the number of processes: 5
enter the arrival time and burst time of 1 process: 1 3
enter the arrival time and burst time of 2 process: 0 4
enter the arrival time and burst time of 3 process: 4 5
enter the arrival time and burst time of 4 process: 3 2
enter the arrival time and burst time of 5 process: 2 6

Pid     AT      BT      ST      CT      TAT     WT
1       1       3       6       9       8       5
2       0       4       0       4       4       0
3       4       5       9       14      10      5
4       3       2       4       6       3       1
5       2       6       14      20      18      12
Average Turnaround Time = 8.60
Average Waiting Time = 4.60
CPU Utilization = 100.00%
Throughput = 0.25 processes/unit time
PS C:\Users\91962\Desktop\shubhamD>
```

```
PS C:\Users\91962\Desktop\shubhamD> cd "c:\Users\91962\Desktop\shubhamD\" ;
Enter the number of processes: 5
enter the arrival time and burst time of 1 process: 4 3
enter the arrival time and burst time of 2 process: 2 1
enter the arrival time and burst time of 3 process: 5 4
enter the arrival time and burst time of 4 process: 6 2
enter the arrival time and burst time of 5 process: 1 4

Pid     AT      BT      ST      CT      TAT     WT
1       4       3       8       11      7       4
2       2       1       2       3       1       0
3       5       4       11      15      10      6
4       6       2       6       8       2       0
5       1       4       1       6       5       1
Average Turnaround Time = 5.00
Average Waiting Time = 2.20
CPU Utilization = 93.33%
Throughput = 0.36 processes/unit time
PS C:\Users\91962\Desktop\shubhamD>
```