Operating systems lab

Name :Jagtap Mahesh
Reg No. 24MCS1017

1) Write a C program to support the OS to do the short term scheduling using the following algorithms.
   a) First Come First Serve
   b) Shortest Job First
   c) Shortest Remaining Time First

Process	P1	P2	P3	P4	P5
Arrival Time	0	4	3	6	7
CPU Burst	5	4	7	3	1

Prepare a Gantt chart and calculate the Average Waiting Time and the Turnaround Time. Display which algorithm improves the efficiency for this group of processes.

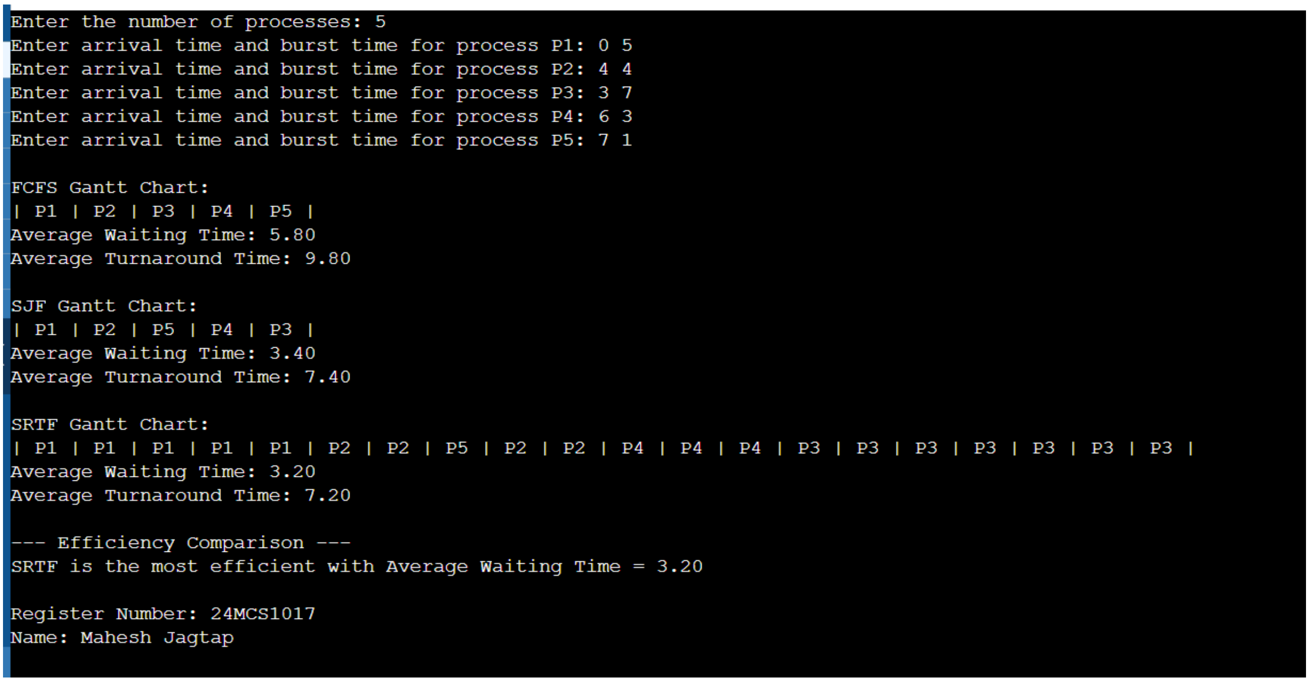
Note: The output must have your register number and name

Code:

```
#include <stdio.h>
#include <limits.h>
float fcfs(int n, int burst[], int arrival[]);
float sjf(int n, int burst[], int arrival[]);
float srtf(int n, int burst[], int arrival[]);
int main() {
  int n;
  printf("Enter the number of processes: ");
  scanf("%d", &n);

  int burst[n], arrival[n];
  for (int i = 0; i < n; i++) {
    printf("Enter arrival time and burst time for process P%d: ", i + 1);
    scanf("%d %d", &arrival[i], &burst[i]);

  }
```

```
    float fcfs_awt = fcfs(n, burst, arrival);
    float sjf_awt = sjf(n, burst, arrival);
    float srtf_awt = srtf(n, burst, arrival);

    printf("\n--- Efficiency Comparison ---\n");
    if (fcfs_awt < sjf_awt && fcfs_awt < srtf_awt) {

        printf("FCFS is the most efficient with Average Waiting Time = %.2f\n", fcfs_awt);
    } else if (sjf_awt < fcfs_awt && sjf_awt < srtf_awt) {
        printf("SJF is the most efficient with Average Waiting Time = %.2f\n", sjf_awt);
    } else {
        printf("SRTF is the most efficient with Average Waiting Time = %.2f\n", srtf_awt);

    }
    printf("\nRegister Number: 24MCS1017 \nName: Mahesh Jagtap\n");
    return 0;
}

float fcfs(int n, int burst[], int arrival[]) {
    int wait[n], tat[n], start[n], total_wt = 0, total_tat = 0, time = 0;
    printf("\nFCFS Gantt Chart:\n|");

    start[0] = arrival[0];
    for (int i = 0; i < n; i++) {
        if (time < arrival[i]) {
            time = arrival[i];
        }
        start[i] = time;
        printf(" P%d |", i + 1);
        time += burst[i];
        tat[i] = time - arrival[i];

        wait[i] = tat[i] - burst[i];

        total_wt += wait[i];

        total_tat += tat[i];

    }


    float avg_wt = (float)total_wt / n;

    printf("\nAverage Waiting Time: %.2f", avg_wt);

    printf("\nAverage Turnaround Time: %.2f\n", (float)total_tat / n);

    return avg_wt;

}


float sjf(int n, int burst[], int arrival[]) {

    int wait[n], tat[n], total_wt = 0, total_tat = 0, completed = 0, time = 0;

    int is_completed[n], min_burst, index;

    for (int i = 0; i < n; i++) is_completed[i] = 0;


    printf("\nSJF Gantt Chart:\n|");


    while (completed != n) {

        min_burst = INT_MAX;

        index = -1;

        for (int i = 0; i < n; i++) {

            if (arrival[i] <= time && !is_completed[i] && burst[i] < min_burst) {

                min_burst = burst[i];

                index = i;

            }

        }

        if (index == -1) {

            time++;

            continue;
        }
        printf(" P%d |", index + 1);
        time += burst[index];
        tat[index] = time - arrival[index];
        wait[index] = tat[index] - burst[index];
        total_wt += wait[index];
        total_tat += tat[index];
        is_completed[index] = 1;
        completed++;
    }
    float avg_wt = (float)total_wt / n;
    printf("\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f\n", (float)total_tat / n);
    return avg_wt;
}

float srtf(int n, int burst[], int arrival[]) {
    int wait[n], tat[n], remaining_burst[n], total_wt = 0, total_tat = 0;
    int completed = 0, time = 0, min_burst, index, finish;
    for (int i = 0; i < n; i++) remaining_burst[i] = burst[i];
    printf("\nSRTF Gantt Chart:\n|");
    while (completed != n) {
        min_burst = INT_MAX;
        index = -1;
        for (int i = 0; i < n; i++) {
            if (arrival[i] <= time && remaining_burst[i] < min_burst && remaining_burst[i] > 0) {
                min_burst = remaining_burst[i];
                index = i;
            }
        }
        if (index == -1) {
            time++;
            continue;

        }
        printf(" P%d |", index + 1);
        remaining_burst[index]--;
        time++;
        if (remaining_burst[index] == 0) {
            completed++;
            finish = time;
            tat[index] = finish - arrival[index];
            wait[index] = tat[index] - burst[index];
            total_wt += wait[index];
            total_tat += tat[index];
        }
    }
    float avg_wt = (float)total_wt / n;
    printf("\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f\n", (float)total_tat / n);
    return avg_wt;
}
```

OUTPUT:



```
Enter the number of processes: 5
Enter arrival time and burst time for process P1: 0 5
Enter arrival time and burst time for process P2: 4 4
Enter arrival time and burst time for process P3: 3 7
Enter arrival time and burst time for process P4: 6 3
Enter arrival time and burst time for process P5: 7 1

FCFS Gantt Chart:
| P1 | P2 | P3 | P4 | P5 |
Average Waiting Time: 5.80
Average Turnaround Time: 9.80

SJF Gantt Chart:
| P1 | P2 | P5 | P4 | P3 |
Average Waiting Time: 3.40
Average Turnaround Time: 7.40

SRTF Gantt Chart:
| P1 | P1 | P1 | P1 | P1 | P2 | P2 | P5 | P2 | P2 | P4 | P4 | P4 | P3 | P3 | P3 | P3 | P3 | P3 | P3 |
Average Waiting Time: 3.20
Average Turnaround Time: 7.20

--- Efficiency Comparison ---
SRTF is the most efficient with Average Waiting Time = 3.20

Register Number: 24MCS1017
Name: Mahesh Jagtap
```