Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.
→ FCFS
→SJF(Pre-emptive and non pre-emptive)
INPUT:

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

void findWaitingTimeFCFS(int processes[], int n, int bt[], int wt[], int at[], int ct[]) {
    for (int i = 0; i < n; i++) {
        wt[i] = ct[i] - at[i] - bt[i];
    }
}

void findWaitingTimeSJFPreemptive(int processes[], int n, int bt[], int wt[], int at[], int ct[]) {
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = bt[i];
    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    while (complete != n) {
        for (int j = 0; j < n; j++) {
            if ((at[j] <= t) && (rt[j] < minm) && (rt[j] > 0)) {
                minm = rt[j];
                shortest = j;
            }
        }
        rt[shortest]--;
        minm = rt[shortest];
        if (minm == 0)
            minm = INT_MAX;
        if (rt[shortest] == 0) {
            complete++;
            finish_time = t + 1;
            wt[shortest] = finish_time - bt[shortest] - at[shortest];
            if (wt[shortest] < 0)
                wt[shortest] = 0;
            ct[shortest] = finish_time;
        }
        t++;
    }
}


void findWaitingTimeSJFNonPreemptive(int processes[], int n, int bt[], int wt[], int at[], int ct[]) {
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = bt[i];

    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    while (complete != n) {
        for (int j = 0; j < n; j++) {
```

```
                minm = rt[j];
                shortest = j;
            }
        }
        t += rt[shortest];
        finish_time = t;
        wt[shortest] = finish_time - bt[shortest] - at[shortest];
        if (wt[shortest] < 0)
            wt[shortest] = 0;
        rt[shortest] = INT_MAX;
        complete++;
        ct[shortest] = finish_time;
        minm = INT_MAX;
    }
}


void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[], int ct[], int at[]) {
    for (int i = 0; i < n; i++)
        tat[i] = ct[i] - at[i];
}
void findAverageTimeFCFS(int processes[], int n, int bt[], int at[], int ct[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;

    findWaitingTimeFCFS(processes, n, bt, wt, at, ct);
    findTurnAroundTime(processes, n, bt, wt, tat, ct, at);

    printf("FCFS Scheduling\n");
    printf("Processes Arrival time Burst time Waiting time Turn around time Completion time\n");

    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf(" %d ", processes[i]);
        printf("         %d ", at[i]);
        printf("         %d ", bt[i]);
        printf("          %d", wt[i]);
        printf("                  %d", tat[i]);
        printf("                  %d\n", ct[i]);
    }

    float avg_wt = (float)total_wt / n;
    float avg_tat = (float)total_tat / n;
    printf("Average waiting time = %f\n", avg_wt);
```

```
    printf("Average turn around time = %f\n", avg_tat);
}

void findAverageTimeSJFNonPreemptive(int processes[], int n, int bt[], int at[], int ct[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;

    findWaitingTimeSJFNonPreemptive(processes, n, bt, wt, at, ct);
    findTurnAroundTime(processes, n, bt, wt, tat, ct , at);

    printf("\nSJF (Non-preemptive) Scheduling\n");
    printf("Processes Arrival time Burst time Waiting time Turn around time Completion time\n");

    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf(" %d ", processes[i]);
        printf("         %d ", at[i]);
        printf("         %d ", bt[i]);
        printf("          %d", wt[i]);
        printf("                  %d", tat[i]);
        printf("                  %d\n", ct[i]);
    }
```

```
    float avg_tat = (float)total_tat / n;
    printf("Average waiting time = %f\n", avg_wt);
    printf("Average turn around time = %f\n", avg_tat);
}


void findAverageTimeSJFPreemptive(int processes[], int n, int bt[], int at[], int ct[]) {
    int wt[n], tat[n];
    int total_wt = 0, total_tat = 0;

    findWaitingTimeSJFPreemptive(processes, n, bt, wt, at, ct);
    findTurnAroundTime(processes, n, bt, wt, tat, ct , at);

    printf("\nSJF (Preemptive) Scheduling\n");
    printf("Processes Arrival time Burst time Waiting time Turn around time Completion time\n");

    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf(" %d ", processes[i]);
        printf("         %d ", at[i]);
        printf("         %d ", bt[i]);
        printf("          %d", wt[i]);
        printf("                  %d", tat[i]);
```

```
    float avg_tat = (float)total_tat / n;
    printf("Average waiting time = %f\n", avg_wt);
    printf("Average turn around time = %f\n", avg_tat);
}

int main() {
    int processes[10], burst_time[10], arrival_time[10], completion_time[10];
    int n;

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

    printf("Enter arrival time and burst time for each process:\n");
    for (int i = 0; i < n; i++) {
        printf("Arrival time of process[%d]: ", i + 1);
        scanf("%d", &arrival_time[i]);
        printf("Burst time of process[%d]: ", i + 1);
        scanf("%d", &burst_time[i]);
        processes[i] = i + 1;
    }

    completion_time[0] = arrival_time[0] + burst_time[0];
    for (int i = 1; i < n; i++) {
        if (arrival_time[i] > completion_time[i - 1]) {
```

```
    for (int i = 0; i < n; i++) {
        printf("Arrival time of process[%d]: ", i + 1);
        scanf("%d", &arrival_time[i]);
        printf("Burst time of process[%d]: ", i + 1);
        scanf("%d", &burst_time[i]);
        processes[i] = i + 1;
    }

    completion_time[0] = arrival_time[0] + burst_time[0];
    for (int i = 1; i < n; i++) {
        if (arrival_time[i] > completion_time[i - 1]) {
            completion_time[i] = arrival_time[i] + burst_time[i];
        } else {
            completion_time[i] = completion_time[i - 1] + burst_time[i];
        }
    }

    findAverageTimeFCFS(processes, n, burst_time, arrival_time, completion_time);
    findAverageTimeSJFNonPreemptive(processes, n, burst_time, arrival_time, completion_time);
    findAverageTimeSJFPreemptive(processes, n, burst_time, arrival_time, completion_time);

    return 0;
}
```

OUTPUT:

```
Enter the number of processes: 3
Enter arrival time and burst time for each process:
Arrival time of process[1]: 0
Burst time of process[1]: 2
Arrival time of process[2]: 1
Burst time of process[2]: 4
Arrival time of process[3]: 2
Burst time of process[3]: 3
FCFS Scheduling
Processes Arrival time Burst time Waiting time Turn around time Completion time
 1          0          2          0                2                   2
 2          1          4          1                5                   6
 3          2          3          4                7                   9
Average waiting time = 1.666667
Average turn around time = 4.666667

SJF (Non-preemptive) Scheduling
Processes Arrival time Burst time Waiting time Turn around time Completion time
 1          0          2          0                2                   2
 2          1          4          4                8                   9
 3          2          3          0                3                   5
Average waiting time = 1.333333
Average turn around time = 4.333333

SJF (Preemptive) Scheduling
Processes Arrival time Burst time Waiting time Turn around time Completion time
 1          0          2          0                2                   2
 2          1          4          4                8                   9
 3          2          3          0                3                   5
Average waiting time = 1.333333
Average turn around time = 4.333333
```