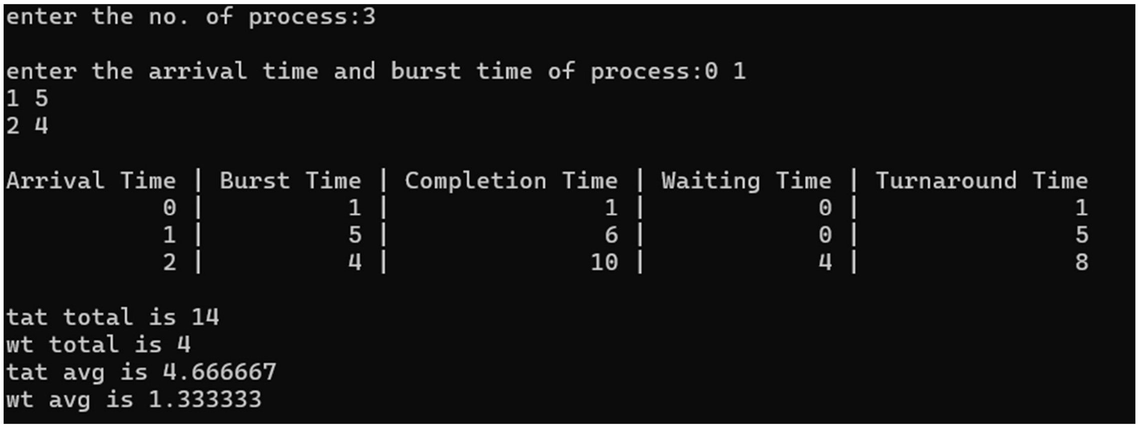
write c program to simulate cpu scheduling algorithm

1. FCFS
2. SJF

code:

```
#include<stdio.h>
#include<stdlib.h>
int at[10],bt[10],ct[10],tat[10],wt[10],rt[10];
int i,j,k,n,temp,temp1;
void main()
{
   printf("enter the no. of process:");
   scanf("%d",&n);
   printf("\nenter the arrival time and burst time of process:");
   for(i=0;i<n;i++){
       scanf("%d",&at[i]);
       scanf("%d",&bt[i]);
   }
   for(i=0;i<n;i++){
for(j=i+1;j<n;j++) {
           if(at[j]<at[i]) {
             temp=at[i];
             at[i]=at[j];
             at[j]=temp;
             temp1=bt[i];
             bt[i]=bt[j];
             bt[j]=temp1;
           }
       }
   }
   int ct1=0;
   for(i=0;i<n;i++){
       if(ct1<at[i]) {
           ct1=at[i];
       }
         ct[i]=ct1+bt[i];
       ct1=ct[i];
   }
    for(i=0;i<n;i++)
   {
       tat[i]=ct[i]-at[i];
       wt[i]=tat[i]-bt[i];


    }
    int t_tat=0, t_wt;
    for(i=0;i<n;i++)
   {
     t_tat+=tat[i];
     t_wt+=wt[i];
    }
printf("\nArrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%12d | %10d | %15d | %12d | %15d\n", at[i], bt[i], ct[i], wt[i], tat[i] );
    }
  float tat_avg=(float)t_tat/n;
    float wt_avg=(float)t_wt/n;
    printf("\ntat total is %d",t_tat );
    printf("\nwt total is %d",t_wt);
    printf("\ntat avg is %f", tat_avg);
    printf("\nwt avg is %f\n", wt_avg);
}
```



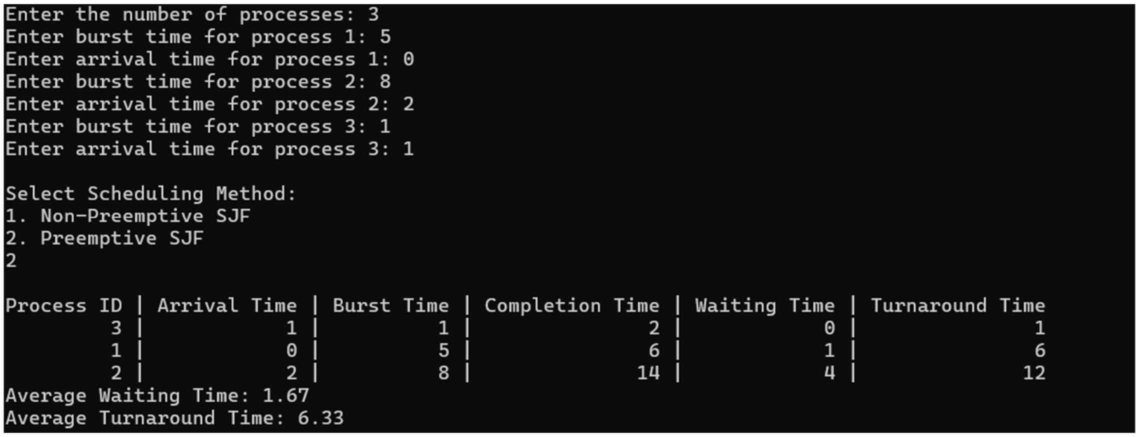
```
enter the no. of process:3

enter the arrival time and burst time of process:0 1
1 5
2 4

Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time
           0 |          1 |               1 |            0 |               1
           1 |          5 |               6 |            0 |               5
           2 |          4 |              10 |            4 |               8

tat total is 14
wt total is 4
tat avg is 4.666667
wt avg is 1.333333
```

```
#include <stdio.h>
#include <stdbool.h>

struct Process {
    int id;
    int burst_time;
    int arrival_time;
    int remaining_time;
    int waiting_time;
    int turnaround_time;
    int completion_time;
};
void findWaitingTimeNonPreemptive(struct Process proc[], int n) {
    int total_time = 0;
    proc[0].waiting_time = 0;
    total_time += proc[0].burst_time;
 for (int i = 1; i < n; i++) {
        proc[i].waiting_time = proc[i-1].waiting_time + proc[i-1].burst_time;
        total_time += proc[i].burst_time;
    }
}

void findCompletionTimeNonPreemptive(struct Process proc[], int n) {
    for (int i = 0; i < n; i++) {
        proc[i].completion_time = proc[i].waiting_time + proc[i].burst_time;
    }
}
void findTurnaroundTimeNonPreemptive(struct Process proc[], int n) {
    for (int i = 0; i < n; i++) {
        proc[i].turnaround_time = proc[i].completion_time - proc[i].arrival_time;
    }
}

void findWaitingTimePreemptive(struct Process proc[], int n) {
    int completed = 0, time = 0;
    int min_remaining_time, shortest = -1;
while (completed < n) {
        min_remaining_time = 10000;

        for (int i = 0; i < n; i++) {
            if (proc[i].arrival_time <= time && proc[i].remaining_time > 0 && proc[i].remaining_time < min_remaining_time) {
                min_remaining_time = proc[i].remaining_time;
                shortest = i;
            }
        }
if (shortest == -1) {
            time++;
            continue;
        }

        proc[shortest].remaining_time--;
        time++;

        if (proc[shortest].remaining_time == 0) {
            proc[shortest].completion_time = time;
            completed++;
        }
    }
}

void findTurnaroundTimePreemptive(struct Process proc[], int n) {
    for (int i = 0; i < n; i++) {
        proc[i].turnaround_time = proc[i].completion_time - proc[i].arrival_time;
    }
}

void findWaitingTimeFinal(struct Process proc[], int n) {
    for (int i = 0; i < n; i++) {
        proc[i].waiting_time = proc[i].turnaround_time - proc[i].burst_time;
    }
```

```
}

void findAverageTime(struct Process proc[], int n) {
    int total_waiting_time = 0, total_turnaround_time = 0;

    for (int i = 0; i < n; i++) {
        total_waiting_time += proc[i].waiting_time;
        total_turnaround_time += proc[i].turnaround_time;
    }

    printf("Average Waiting Time: %.2f\n", (float)total_waiting_time / n);
    printf("Average Turnaround Time: %.2f\n", (float)total_turnaround_time / n);
}

void sortByBurstTime(struct Process proc[], int n) {
    struct Process temp;

    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (proc[i].burst_time > proc[j].burst_time) {
                temp = proc[i];
                proc[i] = proc[j];
                proc[j] = temp;
            }
        }
    }
}

void printProcessDetails(struct Process proc[], int n) {
    printf("\nProcess ID | Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%10d | %12d | %10d | %15d | %12d | %15d\n", proc[i].id, proc[i].arrival_time, proc[i].burst_time,
proc[i].completion_time, proc[i].waiting_time, proc[i].turnaround_time);
    }
}
```

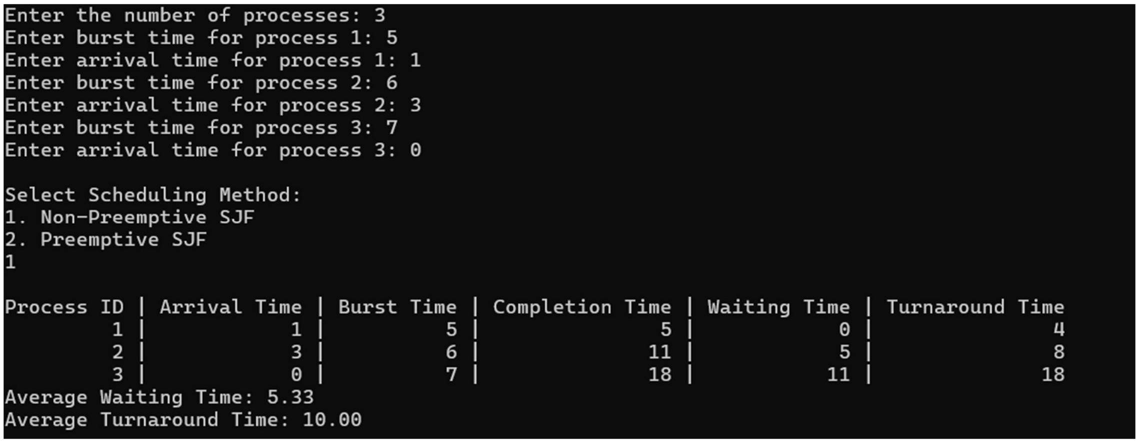
```
int main() {
    int n, choice;

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

    struct Process proc[n];

    for (int i = 0; i < n; i++) {
        proc[i].id = i + 1;
        printf("Enter burst time for process %d: ", proc[i].id);
        scanf("%d", &proc[i].burst_time);
        proc[i].remaining_time = proc[i].burst_time;
        printf("Enter arrival time for process %d: ", proc[i].id);
        scanf("%d", &proc[i].arrival_time);
    }

    printf("\nSelect Scheduling Method:\n1. Non-Preemptive SJF\n2. Preemptive SJF\n");
    scanf("%d", &choice);

    if (choice == 1) {

        sortByBurstTime(proc, n);
        findWaitingTimeNonPreemptive(proc, n);
        findCompletionTimeNonPreemptive(proc, n);
        findTurnaroundTimeNonPreemptive(proc, n);
        printProcessDetails(proc, n);
    }
    else if (choice == 2) {

        sortByBurstTime(proc, n);
        findWaitingTimePreemptive(proc, n);
        findTurnaroundTimePreemptive(proc, n);
        findWaitingTimeFinal(proc,n);
        printProcessDetails(proc, n);
    }
    else {
        printf("Invalid choice.\n");
        return 1;
    }



    findAverageTime(proc, n);


    return 0;
}
```

PREEMTIVE:



```
Enter the number of processes: 3
Enter burst time for process 1: 5
Enter arrival time for process 1: 0
Enter burst time for process 2: 8
Enter arrival time for process 2: 2
Enter burst time for process 3: 1
Enter arrival time for process 3: 1

Select Scheduling Method:
1. Non-Preemptive SJF
2. Preemptive SJF
2

Process ID | Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time
         3 |            1 |          1 |               2 |            0 |               1
         1 |            0 |          5 |               6 |            1 |               6
         2 |            2 |          8 |              14 |            4 |              12
Average Waiting Time: 1.67
Average Turnaround Time: 6.33
```

NON-PREEMTIVE:



```
Enter the number of processes: 3
Enter burst time for process 1: 5
Enter arrival time for process 1: 1
Enter burst time for process 2: 6
Enter arrival time for process 2: 3
Enter burst time for process 3: 7
Enter arrival time for process 3: 0

Select Scheduling Method:
1. Non-Preemptive SJF
2. Preemptive SJF
1

Process ID | Arrival Time | Burst Time | Completion Time | Waiting Time | Turnaround Time
         1 |            1 |          5 |               5 |            0 |               4
         2 |            3 |          6 |              11 |            5 |               8
         3 |            0 |          7 |              18 |           11 |              18
Average Waiting Time: 5.33
Average Turnaround Time: 10.00
```