PROGRAM 1: FCFS:

CODE:

```
#include<stdio.h>

int main(){

int n,arrv_t[30],bur_t[30],com_t[30],wait_t[30],tat[30],temp=0;
float avg_wt_t=0,avg_tat_t=0;

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

printf("\nEnter the arrival time:");
for(int i=0;i<n;i++)
{
 printf("P_id[%d]:",i+1);
       scanf("%d",&arrv_t[i]);
}
printf("\nEnter the burst time:");
for(int i=0;i<n;i++)
{
   printf("P_id[%d]:",i+1);
    scanf("%d",&bur_t[i]);
}
 for(int i=0;i<n;i++)
 {
     if(arrv_t[i]<=temp)
     {
        com_t[i] = temp + bur_t[i];
     }
   else
     {
        com_t[i] = arrv_t[i] + bur_t[i];
        temp = arrv_t[i];
     }

     tat[i] = com_t[i] - arrv_t[i];
     wait_t[i] = tat[i] - bur_t[i];
```

```
    temp = com_t[i];

    avg_tat_t += tat[i];
    avg_wt_t += wait_t[i];
 }
    printf("\nProcess id \t Arrival time \t Burst time \t Compile time\t TurnAround time \t Waiting time\n");
    for(int i=0;i<n;i++)
    {
       printf("\nP_id%d \t\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n",i,arrv_t[i],bur_t[i],com_t[i],tat[i],wait_t[i]);
    }

   avg_tat_t /= n;
   avg_wt_t /= n;

   printf("\n Average turnAround time = %f",avg_tat_t);
   printf("\n Average waiting time = %f",avg_wt_t);

   return 0;

}
```

OUTPUT:

```
Enter the number of processes:4

Enter the arrival time:P_id[1]:0
P_id[2]:1
P_id[3]:5
P_id[4]:6

Enter the burst time:P_id[1]:2
P_id[2]:2
P_id[3]:3
P_id[4]:4

Process id       Arrival time    Burst time      Compile time    TurnAround time         Waiting time

P_id0                    0              2              2              2              0

P_id1                    1              2              4              3              1

P_id2                    5              3              8              3              0

P_id3                    6              4              12             6              2

 Average turnAround time = 3.500000
 Average waiting time = 0.750000
Process returned 0 (0x0)   execution time : 20.850 s
Press any key to continue.
```

PROGRAM 2: SJF

CODE:

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

struct Process {
    int process_id;
    int burst_time;
    int arrival_time;
    int waiting_time;
    int turnaround_time;
};

void sort_by_arrival_time(struct Process *processes, int n) {
    struct Process temp;
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (processes[j].arrival_time > processes[j + 1].arrival_time) {
                temp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = temp;
            }
        }
    }
}

void sjf(struct Process *processes, int n) {
    sort_by_arrival_time(processes, n);
    int total_waiting_time = 0, total_turnaround_time = 0;
    int *remaining_burst_time = (int *)malloc(n * sizeof(int));
    int completed = 0, current_time = 0;
    for (int i = 0; i < n; i++) {
        remaining_burst_time[i] = processes[i].burst_time;
    }
    while (completed < n) {
        int shortest_burst_index = -1;
```

```
        int shortest_burst = 9999; // A large value to find minimum
        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= current_time && remaining_burst_time[i] <
shortest_burst && remaining_burst_time[i] > 0) {
                shortest_burst = remaining_burst_time[i];
                shortest_burst_index = i;
            }
        }
        if (shortest_burst_index == -1) {
            current_time++;
            continue;
        }
        remaining_burst_time[shortest_burst_index]--;
        if (remaining_burst_time[shortest_burst_index] == 0) {
            completed++;
            processes[shortest_burst_index].turnaround_time = current_time + 1 -
processes[shortest_burst_index].arrival_time;
            processes[shortest_burst_index].waiting_time =
processes[shortest_burst_index].turnaround_time - processes[shortest_burst_index].burst_time;
            total_waiting_time += processes[shortest_burst_index].waiting_time;
            total_turnaround_time += processes[shortest_burst_index].turnaround_time;
        }
        current_time++;
    }
    printf("\nProcess ID\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\n", processes[i].process_id, processes[i].burst_time,
                processes[i].waiting_time, processes[i].turnaround_time);
    }
    printf("\nAverage Waiting Time: %.2f", (float)total_waiting_time / n);
    printf("\nAverage Turnaround Time: %.2f", (float)total_turnaround_time / n);
    free(remaining_burst_time);
}

int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process *processes = (struct Process *)malloc(n * sizeof(struct Process));
    printf("Enter the process id, burst time, and arrival time for each process:\n");
```

```
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d %d %d", &processes[i].process_id, &processes[i].burst_time,
&processes[i].arrival_time);
        processes[i].waiting_time = 0;
        processes[i].turnaround_time = 0;
    }
    sjf(processes, n);
    free(processes);
    return 0;
}
```

OUTPUT:

```
Enter the number of processes: 4
Enter the process id, burst time, and arrival time for each process:
Process 1: 1 6 0
Process 2: 3 7 2
Process 3: 2 3 3
Process 4: 4 5 6

Process ID      Burst Time      Waiting Time    Turnaround Time
1               6               0               6
3               7               12              19
2               3               3               6
4               5               3               8

Average Waiting Time: 4.50
Average Turnaround Time: 9.75
Process returned 0 (0x0)   execution time : 20.170 s
Press any key to continue.
```