Name: Nikhil Gupta
Roll No.: 20051523

SHORTEST JOB FIRST

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

struct process_struct
{
    int pid;
    int at;
    int bt;
    int ct, wt, tat, rt, start_time;
} ps[20];

int findmax(int a, int b)
{
    return a > b ? a : b;
}

int findmin(int a, int b)
{
    return a < b ? a : b;
}

int main()
{

    int n;
    bool is_completed[100] = {false}, is_first_process =
true;
    int current_time = 0;
    int completed = 0;
    printf("Enter total number of processes: ");
    scanf("%d", &n);
    int sum_tat = 0, sum_wt = 0, sum_rt = 0,
total_idle_time = 0, prev = 0, length_cycle;
    float cpu_utilization;
```

Name: Nikhil Gupta
Roll No.: 20051523

```
    int max_completion_time, min_arrival_time;

    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Arrival Time: ", i);
        scanf("%d", &ps[i].at);
        ps[i].pid = i;
    }

    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Burst Time: ", i);
        scanf("%d", &ps[i].bt);
    }

    while (completed != n)
    {

        int min_index = -1;
        int minimum = INT_MAX;
        for (int i = 0; i < n; i++)
        {
            if (ps[i].at <= current_time &&
is_completed[i] == false)
            {
                if (ps[i].bt < minimum)
                {
                    minimum = ps[i].bt;
                    min_index = i;
                }
                if (ps[i].bt == minimum)
                {
                    if (ps[i].at < ps[min_index].at)
                    {
                        minimum = ps[i].bt;
                        min_index = i;
                    }
                }
            }
        }
```

Name: Nikhil Gupta
Roll No.: 20051523

```
        if (min_index == -1)
        {
            current_time++;
        }
        else
        {
            ps[min_index].start_time = current_time;
            ps[min_index].ct = ps[min_index].start_time +
ps[min_index].bt;
            ps[min_index].tat = ps[min_index].ct -
ps[min_index].at;
            ps[min_index].wt = ps[min_index].tat -
ps[min_index].bt;
            ps[min_index].rt = ps[min_index].wt;

            sum_tat += ps[min_index].tat;
            sum_wt += ps[min_index].wt;
            sum_rt += ps[min_index].rt;
            total_idle_time += (is_first_process == true)
? 0 : (ps[min_index].start_time - prev);

            completed++;
            is_completed[min_index] = true;
            current_time = ps[min_index].ct;
            prev = current_time;
            is_first_process = false;
        }
    }

    // Output
    printf("\nProcess No.\tAT\tCPU Burst
Time\tCT\tTAT\tWT\tRT\n");
    for (int i = 0; i < n; i++)
        printf("%d\t\t%d\t%d\t\t%d\t%d\t%d\t%d\n",
ps[i].pid, ps[i].at, ps[i].bt, ps[i].ct, ps[i].tat,
ps[i].wt, ps[i].rt);

    printf("\n");
```

Name: Nikhil Gupta
Roll No.: 20051523

```
    printf("\nAverage Turn Around time= %f ",
(float)sum_tat / n);
    printf("\nAverage Waiting Time= %f ", (float)sum_wt /
n);
    printf("\nAverage Response Time= %f ", (float)sum_rt
/ n);

    return 0;
}
```

OUTPUT

```
PS E:\Mega Sync\Programming\C\Scheduling Algorithms> cd "e:\Mega Sync\
Enter total number of processes: 5

Enter Process 0 Arrival Time: 1

Enter Process 1 Arrival Time: 3

Enter Process 2 Arrival Time: 6

Enter Process 3 Arrival Time: 7

Enter Process 4 Arrival Time: 9

Enter Process 0 Burst Time: 7

Enter Process 1 Burst Time: 3

Enter Process 2 Burst Time: 2

Enter Process 3 Burst Time: 10

Enter Process 4 Burst Time: 8

Process No.     AT      CPU Burst Time  CT      TAT     WT      RT
0               1       7               8       7       0       0
1               3       3               13      10      7       7
2               6       2               10      4       2       2
3               7       10              31      24      14      14
4               9       8               21      12      4       4


Average Turn Around time= 11.400000
Average Waiting Time= 5.400000
Average Response Time= 5.400000
PS E:\Mega Sync\Programming\C\Scheduling Algorithms>
```

Name: Nikhil Gupta
Roll No.: 20051523

SHORTEST REMAINING TIME FIRST

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

struct process_struct
{
    int pid;
    int at;
    int bt;
    int ct, wt, tat, rt, start_time;
} ps[100];

int findmax(int a, int b)
{
    return a > b ? a : b;
}

int findmin(int a, int b)
{
    return a < b ? a : b;
}

int main()
{

    int n;
    float bt_remaining[100];
    bool is_completed[100] = {false}, is_first_process =
true;
    int current_time = 0;
    int completed = 0;
    ;
```

Name: Nikhil Gupta
Roll No.: 20051523

```
    float sum_tat = 0, sum_wt = 0, sum_rt = 0,
total_idle_time = 0, length_cycle, prev = 0;
    float cpu_utilization;

    int max_completion_time, min_arrival_time;

    printf("Enter total number of processes: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Arrival Time: ", i);
        scanf("%d", &ps[i].at);
        ps[i].pid = i;
    }

    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Burst Time: ", i);
        scanf("%d", &ps[i].bt);
        bt_remaining[i] = ps[i].bt;
    }

    while (completed != n)
    {

        int min_index = -1;
        int minimum = INT_MAX;
        for (int i = 0; i < n; i++)
        {
            if (ps[i].at <= current_time &&
is_completed[i] == false)
            {
                if (bt_remaining[i] < minimum)
                {
                    minimum = bt_remaining[i];
                    min_index = i;
                }
                if (bt_remaining[i] == minimum)
                {
                    if (ps[i].at < ps[min_index].at)
```

Name: Nikhil Gupta
Roll No.: 20051523

```
                    {
                        minimum = bt_remaining[i];
                        min_index = i;
                    }
                }
            }
        }

        if (min_index == -1)
        {
            current_time++;
        }
        else
        {
            if (bt_remaining[min_index] ==
ps[min_index].bt)
            {
                ps[min_index].start_time = current_time;
                total_idle_time += (is_first_process ==
true) ? 0 : (ps[min_index].start_time - prev);
                is_first_process = false;
            }
            bt_remaining[min_index] -= 1;
            current_time++;
            prev = current_time;
            if (bt_remaining[min_index] == 0)
            {
                ps[min_index].ct = current_time;
                ps[min_index].tat = ps[min_index].ct -
ps[min_index].at;
                ps[min_index].wt = ps[min_index].tat -
ps[min_index].bt;
                ps[min_index].rt =
ps[min_index].start_time - ps[min_index].at;

                sum_tat += ps[min_index].tat;
                sum_wt += ps[min_index].wt;
                sum_rt += ps[min_index].rt;
                completed++;
                is_completed[min_index] = true;
```

Name: Nikhil Gupta
Roll No.: 20051523

```
            }
        }
    }

    max_completion_time = INT_MIN;
    min_arrival_time = INT_MAX;
    for (int i = 0; i < n; i++)
    {
        max_completion_time =
findmax(max_completion_time, ps[i].ct);
        min_arrival_time = findmin(min_arrival_time,
ps[i].at);
    }
    length_cycle = max_completion_time -
min_arrival_time;

    printf("\nProcess No.\tAT\tCPU Burst
Time\tCT\tTAT\tWT\tRT\n");
    for (int i = 0; i < n; i++)
        printf("%d\t\t%d\t%d\t\t%d\t%d\t%d\t%d\n",
ps[i].pid, ps[i].at, ps[i].bt, ps[i].ct, ps[i].tat,
ps[i].wt, ps[i].rt);

    printf("\n");

    cpu_utilization = (float)(length_cycle -
total_idle_time) / length_cycle;

    printf("\nAverage Turn Around time= %f ",
(float)sum_tat / n);
    printf("\nAverage Waiting Time= %f ", (float)sum_wt /
n);
    printf("\nAverage Response Time= %f ", (float)sum_rt
/ n);
    printf("\nThroughput= %f", n / (float)length_cycle);
    printf("\nCPU Utilization(Percentage)= %f",
cpu_utilization * 100);
    return 0;
}
```

Name: Nikhil Gupta
Roll No.: 20051523

OUTPUT

```
PS E:\Mega Sync\Programming\C\Scheduling Algorithms> cd "e:\Mega Sync
f } ; if ($?) { .\srtf }
Enter total number of processes: 6

Enter Process 0 Arrival Time: 0

Enter Process 1 Arrival Time: 1

Enter Process 2 Arrival Time: 2

Enter Process 3 Arrival Time: 3

Enter Process 4 Arrival Time: 4

Enter Process 5 Arrival Time: 5

Enter Process 0 Burst Time: 8

Enter Process 1 Burst Time: 4

Enter Process 2 Burst Time: 2

Enter Process 3 Burst Time: 1

Enter Process 4 Burst Time: 3

Enter Process 5 Burst Time: 2

Process No.     AT      CPU Burst Time  CT      TAT     WT      RT
0               0       8               20      20      12      0
1               1       4               10      9       5       0
2               2       2               4       2       0       0
3               3       1               5       2       1       1
4               4       3               13      9       6       6
5               5       2               7       2       0       0


Average Turn Around time= 7.333333
Average Waiting Time= 4.000000
Average Response Time= 1.166667
Throughput= 0.300000
CPU Utilization(Percentage)= 100.000000
```