**20BCS042 | MOHD ADIL**

**PROGRAM 4: SRTF**

#include <stdio.h>

struct process

{

    int pid;

    int burst\_time;

    int arrival\_time;

    int waiting\_time;

    int completion\_time;

    int turnaround\_time;

    int response\_time;

    int start\_time;

    int is\_completed;

} pro[100];

int process\_at\_time[100];

void print\_table(int num);

void timeCalculation(int burst\_remaining[], int n);

void sortCompletion(int num);

void print\_gantt(int n);

int main()

{

    printf("\n\*\*\*\*\*\*\*\* | 20BCS042| MOHD ADIL | \*\*\*\*\*\*\*\*\n");

    int n;

    int burst\_remaining[100];

    printf("\nEnter the number of processes: ");

    scanf("%d", &n);

    printf("\nEnter the processes:-\n");

    for (int i = 0; i < n; i++)

    {

        printf("\nProcess %d\n", i + 1);

        printf("Arrival Time: ");

        scanf("%d", &pro[i].arrival\_time);

        printf("Burst Time: ");

        scanf("%d", &pro[i].burst\_time);

        pro[i].pid = i + 1;

        burst\_remaining[i] = pro[i].burst\_time;

    }

    timeCalculation(burst\_remaining, n);

    sortCompletion(n);

    print\_gantt(n);

}

void timeCalculation(int burst\_remaining[], int n)

{

    float average\_turnaround\_time;

    float average\_waiting\_time;

    float average\_completion\_time;

    float average\_response\_time;

    float total\_turnaround\_time = 0;

    float total\_waiting\_time = 0;

    float total\_completion\_time = 0;

    float total\_response\_time = 0;

    float total\_idle\_time = 0;

    int current\_time = 0;

    int completed\_pro = 0;

    int prev = 0;

    while (completed\_pro != n)

    {

        int shortest = -1;

        int min = 10000000;

        for (int i = 0; i < n; i++)

        {

            if (pro[i].arrival\_time <= current\_time && pro[i].is\_completed == 0)

            {

                if (burst\_remaining[i] < min)

                {

                    min = burst\_remaining[i];

                    shortest = i;

                }

                else if (burst\_remaining[i] == min)

                {

                    if (pro[i].arrival\_time < pro[shortest].arrival\_time)

                    {

                        min = burst\_remaining[i];

                        shortest = i;

                    }

                }

            }

        }

        if (shortest != -1)

        {

            if (burst\_remaining[shortest] == pro[shortest].burst\_time)

            {

                pro[shortest].start\_time = current\_time;

                total\_idle\_time += pro[shortest].start\_time - prev;

            }

            burst\_remaining[shortest] -= 1;

            current\_time++;

            prev = current\_time;

            if (burst\_remaining[shortest] == 0)

            {

                pro[shortest].completion\_time = current\_time;

                pro[shortest].turnaround\_time = pro[shortest].completion\_time - pro[shortest].arrival\_time;

                pro[shortest].waiting\_time = pro[shortest].turnaround\_time - pro[shortest].burst\_time;

                pro[shortest].response\_time = pro[shortest].start\_time - pro[shortest].arrival\_time;

                total\_turnaround\_time += pro[shortest].turnaround\_time;

                total\_waiting\_time += pro[shortest].waiting\_time;

                total\_response\_time += pro[shortest].response\_time;

                total\_completion\_time += pro[shortest].completion\_time;

                pro[shortest].is\_completed = 1;

                completed\_pro++;

            }

            process\_at\_time[current\_time - 1] = shortest + 1;

        }

        else

        {

            current\_time++;

        }

    }

    average\_waiting\_time = total\_waiting\_time / n;

    average\_response\_time = total\_response\_time / n;

    average\_turnaround\_time = total\_turnaround\_time / n;

    average\_completion\_time = total\_completion\_time / n;

    print\_table(n);

    printf("\nTotal Turnaround Time: %0.2f | Average Turnaround Time: %0.2f", total\_turnaround\_time, average\_turnaround\_time);

    printf("\nTotal Waiting Time:    %0.2f | Average Waiting Time:    %0.2f", total\_waiting\_time, average\_waiting\_time);

    printf("\nTotal Completion Time: %0.2f | Average Completion Time: %0.2f", total\_completion\_time, average\_completion\_time);

    printf("\nTotal Response Time:   %0.2f | Average Response Time:   %0.2f", total\_response\_time, average\_response\_time);

}

void sortCompletion(int num)

{

    struct process temp;

    for (int i = 0; i < num - 1; i++)

    {

        for (int j = 0; j < num - i - 1; j++)

        {

            if (pro[j].completion\_time > pro[j + 1].completion\_time)

            {

                temp = pro[j];

                pro[j] = pro[j + 1];

                pro[j + 1] = temp;

            }

        }

    }

}

void print\_table(int num)

{

    printf("----------------------------------------------------------------------------------------------------------\n");

    printf("| PROCESS | BURST TIME | ARRIVAL TIME | COMPLETION TIME | WAITING TIME | TURNAROUND TIME | RESPONSE TIME |\n");

    printf("----------------------------------------------------------------------------------------------------------\n");

    for (int i = 0; i < num; i++)

    {

        printf("|   P%d    |     %d      |      %d       |       %2d        |      %2d      |       %2d        |      %2d       |\n", pro[i].pid, pro[i].burst\_time, pro[i].arrival\_time, pro[i].completion\_time, pro[i].waiting\_time, pro[i].turnaround\_time, pro[i].response\_time);

        printf("----------------------------------------------------------------------------------------------------------\n");

    }

}

void print\_gantt(int n)

{

    printf("\n\n ---------------------------------------------\n");

    printf("                  GANTT CHART\n");

    printf(" ---------------------------------------------\n");

    printf("\n ");

    for (int i = 0; i < pro[n - 1].completion\_time; i++)

    {

        printf("----");

        printf(" ");

    }

    printf("\n|");

    for (int i = 0; i < pro[n - 1].completion\_time; i++)

    {

        printf(" P%d |", process\_at\_time[i]);

    }

    printf("\n ");

    for (int i = 0; i < pro[n - 1].completion\_time; i++)

    {

        printf("----");

        printf(" ");

    }

    printf("\n");

    for (int i = 0; i <= pro[n - 1].completion\_time; i++)

    {

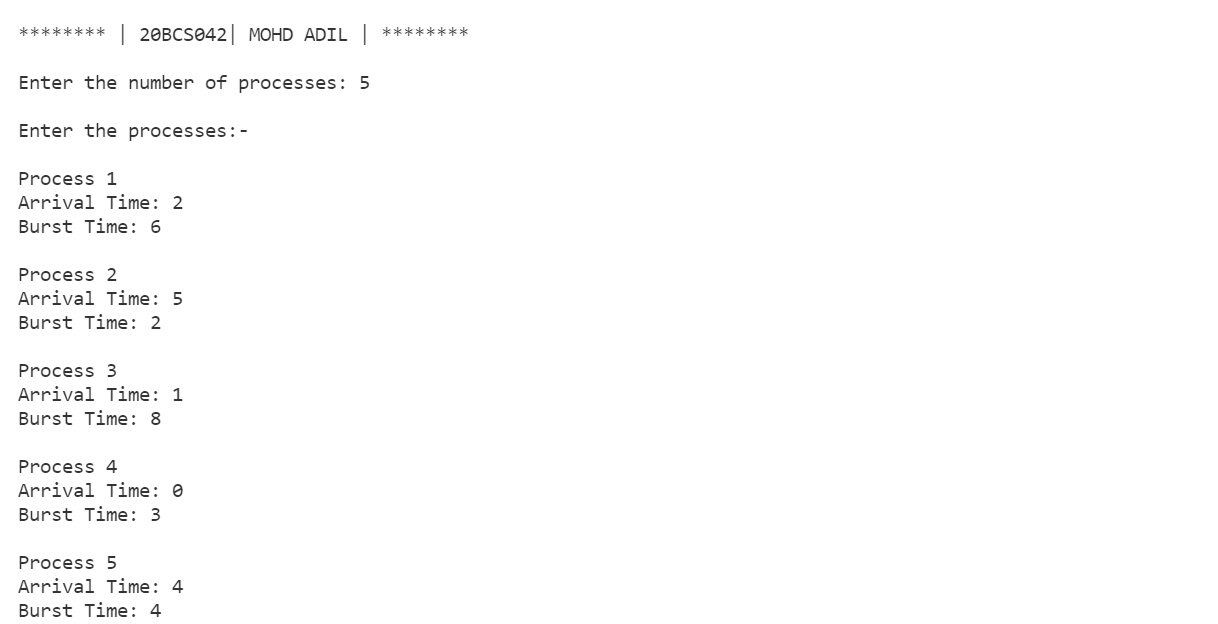
        printf("%2d   ", i);

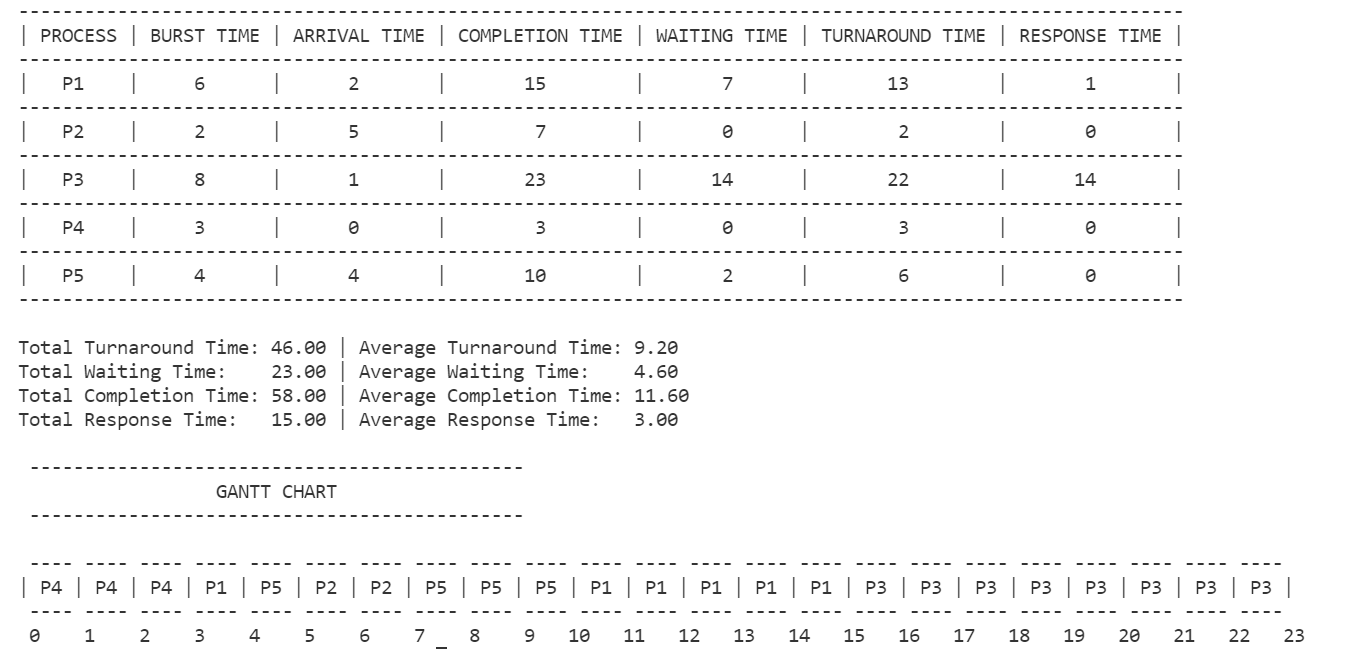
    }

}

//2 6 5 2 1 8 0 3 4 4

**OUTPUT:**

****

****