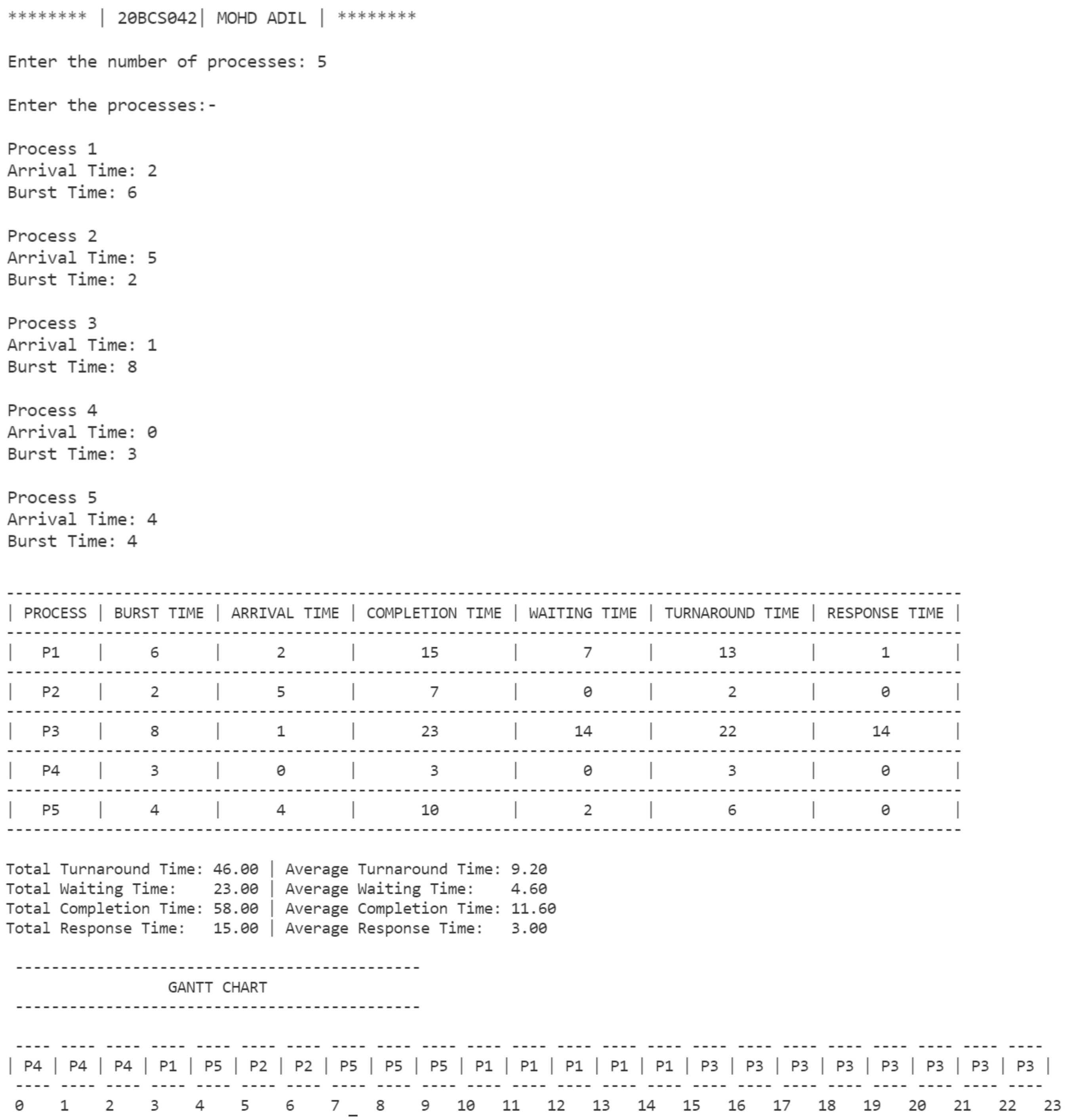
20BCS042 | MOHD ADIL

PROGRAM 4: SRTF

```c
#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:



```
******** | 20BCS042| MOHD ADIL | ********

Enter the number of processes: 5

Enter the processes:-

Process 1
Arrival Time: 2
Burst Time: 6

Process 2
Arrival Time: 5
Burst Time: 2

Process 3
Arrival Time: 1
Burst Time: 8

Process 4
Arrival Time: 0
Burst Time: 3

Process 5
Arrival Time: 4
Burst Time: 4

------------------------------------------------------------------------------------------------
| PROCESS | BURST TIME | ARRIVAL TIME | COMPLETION TIME | WAITING TIME | TURNAROUND TIME | RESPONSE TIME |
------------------------------------------------------------------------------------------------
|   P1    |     6      |      2       |       15        |      7       |       13        |       1       |
------------------------------------------------------------------------------------------------
|   P2    |     2      |      5       |        7        |      0       |        2        |       0       |
------------------------------------------------------------------------------------------------
|   P3    |     8      |      1       |       23        |     14       |       22        |      14       |
------------------------------------------------------------------------------------------------
|   P4    |     3      |      0       |        3        |      0       |        3        |       0       |
------------------------------------------------------------------------------------------------
|   P5    |     4      |      4       |       10        |      2       |        6        |       0       |
------------------------------------------------------------------------------------------------

Total Turnaround Time: 46.00 | Average Turnaround Time: 9.20
Total Waiting Time:    23.00 | Average Waiting Time:    4.60
Total Completion Time: 58.00 | Average Completion Time: 11.60
Total Response Time:   15.00 | Average Response Time:   3.00

 ---------------------------------------------
                 GANTT CHART
 ---------------------------------------------

 ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
| P4 | P4 | P4 | P1 | P5 | P2 | P2 | P5 | P5 | P5 | P1 | P1 | P1 | P1 | P1 | P3 | P3 | P3 | P3 | P3 | P3 | P3 | P3 |
 ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
 0    1    2    3    4    5    6    7 _  8    9    10   11   12   13   14   15   16   17   18   19   20   21   22   23
```