FAIZAN CHOUDHARY

20BCS021

OS LAB

10th February 2022

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <string.h>
#include <limits.h>
using namespace std;

struct process
{
    char n[10];
    int burst;
    int arrival;
    int start;
    int completion;
    int waiting;
    int turnaround;
    int response;
};
process pr[100];
int n;                      // no of processes input from user

struct Gantt
{
    int idx;
    int start;
    int end;
};
Gantt g[100];
int count = 0;                  // to count number of indexed processes for Gantt chart

int remaining[100];         // to store remaining burst time for each process
bool completed[100];        // to store if the process is completed or not
int current_time = 0;
int num = 0;                // to store the number of processes completed

double tot_ct = 0, tot_wt =0, tot_tat = 0, tot_rt =0;

void SRTF () {
    int temp=-1;
    while (num != n) {
        int index = -1;             // to store the index of the process with minimum
burst time
        int mn = INT_MAX;           // to store minimum burst time from all processes
        for (int i=0; i<n; i++) {
```

```
            if (pr[i].arrival <= current_time && completed[i] == false) {
                // if there exists a process with burst time lower than mn, update mn
                if (remaining[i] < mn) {
                    mn = remaining[i];
                    index = i;
                }
                // if two processes have the same burst time, priority given to the one
arriving first
                if (remaining[i] == mn) {
                    if (pr[i].arrival < pr[index].arrival) {
                        mn = remaining[i];
                        index = i;
                    }
                }
            }
        }

        // if it is a valid index, processes present in ready queue
        if (index != -1) {
            // if the burst time matches with the remaining burst time, the process starts
executing for the first time
            if (remaining[index] == pr[index].burst)
                pr[index].start = current_time;

            // updating remaining burst time with a time quantum of 1 unit, and increasing
current time
            remaining[index] -= 1;
            current_time++;

            if (remaining[index] == 0) {
                pr[index].completion = current_time;
                pr[index].turnaround = pr[index].completion - pr[index].arrival;
                pr[index].waiting = pr[index].turnaround - pr[index].burst;
                pr[index].response = pr[index].start - pr[index].arrival;

                tot_tat += pr[index].turnaround;
                tot_wt += pr[index].waiting;
                tot_ct += pr[index].completion;
                tot_rt += pr[index].response;

                completed[index] = true;
                num++;
            }

            // printing Gantt chart
            // cout<<"|"<<current_time-1<<"  "<<pr[index].n<<"  "<<current_time<<"|  ";
            g[count].idx = index;
            g[count].start = current_time - 1;
            g[count].end = current_time;
            count++;

        }
        // if no process in ready queue, increase current_time
        else
```

```
            current_time++;
        }
        g[count].end = current_time;
}

void display () {
    int time = 0;
    cout<<"\n\nProcess | Burst Time | Arrival Time | Completion Time | Waiting Time |
Turnaround Time | Response Time\n";
    cout<<"_____________________________________________________________________________
_____________________\n\n";

    for (int i=0; i<n; i++) {
        printf("   %s        %2d            %2d                %2d               %2d
      %2d              %2d\n", pr[i].n, pr[i].burst, pr[i].arrival, pr[i].completion,
pr[i].waiting, pr[i].turnaround, pr[i].response);
    }

    cout<<"_____________________________________________________________________________
_____________________\n\n";

    printf("\nAverage Completion time: %.2f",tot_ct / (float) n);
    printf("\nAverage Waiting time: %.2f", tot_wt / (float) n);
    printf("\nAverage Turnaround time: %.2f",tot_tat / (float) n);
    printf("\nAverage Response time: %.2f\n",tot_rt / (float) n);
}

void displayGantt () {
    cout<<"\nGantt chart: \n";
    int time = 0;
    for (int i=0; i<count; i++) {
        cout<<"|";
        for (int j=-1; j <= (g[i].start-g[i].end); j++)
            cout<<" ";
        cout<<pr[g[i].idx].n;
        for (int j=-1; j <= (g[i].start-g[i].end); j++)
            cout<<" ";
    }
    cout<<"|\n";
    int i;
    for (i=0; i<count; i++) {
        if (g[i].start > 9)
            cout<<g[i].start<<"   ";
        else if (g[i].start <= 9)
            cout<<g[i].start<<"    ";
    }
    cout<<g[i].end<<endl;

}

int main () {
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nShortest Job First (Preemptive) / Shortest Remaining Time First Scheduling
Algorithm\n";
```

```
    cout<<"\nEnter the number of processes: ";
    cin>>n;
    char k[n][10];
    int bt[n], at[n];                   // burst time and arrival time

    cout<<"\nEnter process names: ";
    for (int i=0; i<n; i++)
        cin>>k[i];
    cout<<"\nEnter burst time for each process: ";
    for (int i=0; i<n; i++)
        cin>>bt[i];
    cout<<"\nEnter arrival time for each process: ";
    for (int i=0; i<n; i++)
        cin>>at[i];

    for (int i=0; i<n; i++) {
        strcpy(pr[i].n, k[i]);
        pr[i].arrival = at[i];
        pr[i].burst = bt[i];
        remaining[i] = pr[i].burst;
    }

    SRTF ();            // logic for calculating various times
    display ();         // displaying calculated values of time
    displayGantt ();    // printing Gantt chart

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Shortest Job First (Preemptive) / Shortest Remaining Time First Scheduling Algorithm

Enter the number of processes: 5

Enter process names: p1 p2 p3 p4 p5

Enter burst time for each process: 6 2 8 3 4

Enter arrival time for each process: 2 5 1 0 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
--------------------------------------------------------------------------------------------------------


Average Completion time: 11.60
Average Waiting time: 4.60
Average Turnaround time: 9.20
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
```

```
FAIZAN CHOUDHARY
20BCS021

Shortest Job First (Preemptive) / Shortest Remaining Time First Scheduling Algorithm

Enter the number of processes: 5

Enter process names: p1 p2 p3 p4 p5

Enter burst time for each process: 3 7 4 2 2

Enter arrival time for each process: 0 6 6 6 5
```

```
Process | Burst Time | Arrival Time | Completion Time | Waiting Time | Turnaround Time | Response Time
--------------------------------------------------------------------------------------------------------

   p1         3            0               3              0              3             0
   p2         7            6              20              7             14             7
   p3         4            6              13              3              7             3
   p4         2            6               9              1              3             1
   p5         2            5               7              0              2             0
--------------------------------------------------------------------------------------------------------


Average Completion time: 10.40
Average Waiting time: 2.20
Average Turnaround time: 5.80
Average Response time: 2.20

Gantt chart:
| p1 | p1 | p1 | p5 | p5 | p4 | p4 | p3 | p3 | p3 | p3 | p2 | p2 | p2 | p2 | p2 | p2 | p2 |
0    1    2    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20
```

```
FAIZAN CHOUDHARY
20BCS021

Shortest Job First (Preemptive) / Shortest Remaining Time First Scheduling Algorithm

Enter the number of processes: 4

Enter process names: p1 p2 p3 p4

Enter burst time for each process: 7 4 1 4

Enter arrival time for each process: 0 2 4 5
```

```
Process | Burst Time | Arrival Time | Completion Time | Waiting Time | Turnaround Time | Response Time
_____________________________________________________________________________________________________

   p1         7            0              16              9             16              0
   p2         4            2               7              1              5              0
   p3         1            4               5              0              1              0
   p4         4            5              11              2              6              2
_____________________________________________________________________________________________________


Average Completion time: 9.75
Average Waiting time: 3.00
Average Turnaround time: 7.00
Average Response time: 0.50

Gantt chart:
| p1 | p1 | p2 | p2 | p3 | p2 | p2 | p4 | p4 | p4 | p4 | p1 | p1 | p1 | p1 | p1 |
0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16
```