FAIZAN CHOUDHARY

20BCS021

OS LAB

10th March 2022

CODE: (code pasted in this format for readability)

```
#include <iostream>
#include <algorithm>
#include <limits.h>
using namespace std;
const int SIZE = 50;

struct process
{
    int pid;
    int burst;
    int arrival;
    int start;
    int completion;
    int waiting;
    int turnaround;
    int response;
};
process pr[SIZE];
int n;

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

int current_time = 0;
bool completed[SIZE] = {false};        // to store if the process is completed or not
int idx = -1;
int num = 0;                           // to store the number of processes completed
int tot_bt = 0;
double mx = -1.0;                       // to store the max response ratio

double tot_ct = 0, tot_wt =0, tot_tat = 0, tot_rt =0;
double hrrn[SIZE];                      // to store the response ratios
double RR;

// comparing wrt arrival time
```

```
bool compare1 (process &p1, process &p2) {
    return p1.arrival < p2.arrival;
}

// comparing wrt pid
bool compare2 (process &p1, process &p2) {
    return p1.pid < p2.pid;
}

void HRRN () {
    sort(pr,pr+n,compare1);
    if (current_time < pr[0].arrival)
        current_time = pr[0].arrival;
    while (num < n) {
        for (int i=0; i<n; i++) {
            RR = ((double)(current_time - pr[i].arrival + pr[i].burst)) / ((double)
pr[i].burst);

            if (RR == mx) {
                if (pr[i].arrival < pr[idx].arrival)
                    idx = i;
            }

            if (RR > mx) {
                if (pr[i].arrival <= current_time && completed[i] == false) {
                    mx = RR;
                    idx = i;
                }
            }
        }

        if (idx != -1) {
            pr[idx].start = current_time;
            pr[idx].completion = pr[idx].start + pr[idx].burst;
            pr[idx].turnaround = pr[idx].completion - pr[idx].arrival;
            pr[idx].waiting = pr[idx].turnaround - pr[idx].burst;
            pr[idx].response = pr[idx].start - pr[idx].arrival;

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

            completed[idx] = true;
            num++;
            current_time = pr[idx].completion;
        }

        else
            current_time++;

        // for Gantt chart
        g[cnt].idx = idx;
        g[cnt].start = pr[idx].start;
```

```
        g[cnt].end = pr[idx].completion;
        cnt++;
    }
    g[cnt].end = current_time;
}

void display () {
    int time = 0;
    // sort(pr,pr+n,compare2);
    process k[SIZE];
    for (int i=0; i<n; i++)
        k[i] = pr[i];
    sort(k,k+n,compare2);

    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("   P%d        %2d            %2d               %2d               %2d
        %2d              %2d\n", k[i].pid, k[i].burst, k[i].arrival, k[i].completion,
k[i].waiting, k[i].turnaround, k[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;
    // if (time < pr[g[0].idx].arrival)
    //     time = pr[g[0].idx].arrival;
    for (int i=0; i<cnt; i++) {
        cout<<"| ";
        cout<<"P"<<pr[g[i].idx].pid<<" ";
    }
    cout<<"|\n";
    int i;
    for (i=0; i<cnt; 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;
}
```

```cpp
int main () {

    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nHighest Response Ratio Next Scheduling Algorithm\n";

    cout<<"\nEnter the number of processes: ";
    cin>>n;

    int *bt = new int[n];
    int *at = new int[n];                    // burst time and arrival time

    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++) {
        pr[i].pid = i+1;
        pr[i].arrival = at[i];
        pr[i].burst = bt[i];
        // bt_copy[i] = bt[i];
        tot_bt += bt[i];
    }

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

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Highest Response Ratio Next Scheduling Algorithm

Enter the number of processes: 5

Enter burst time for each process: 3 6 8 4 5

Enter arrival time for each process: 1 3 5 7 8
```

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

   P1        3            1              4              0             3              0
   P2        6            3             10              1             7              1
   P3        8            5             27             14            22             14
   P4        4            7             14              3             7              3
   P5        5            8             19              6            11              6
--------------------------------------------------------------------------------------------------------


Average Completion time: 14.80
Average Waiting time: 4.80
Average Turnaround time: 10.00
Average Response time: 4.80

Gantt chart:
| P1 | P2 | P4 | P5 | P3 |
1    4    10   14   19   27
```