FAIZAN CHOUDHARY

20BCS021

OS LAB

10th March 2022

CODE: (code pasted in this format for readability)

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

struct process
{
    int pid;
    int priority;
    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 remaining[100];         // to store remaining burst time for each process
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

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

// 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 PrePriorityScheduling () {
    // sort(pr,pr+n,compare1);
    while (num != n) {
        int idx = -1;                   // stores the index of process with highest
priority
        int mn = INT_MAX;               // stores the highest priority (lowest number)
        for (int i=0; i<n; i++) {
            if (pr[i].arrival <= current_time && completed[i] == false) {
                // if a process has greater priority
                if (pr[i].priority < mn) {
                    mn = pr[i].priority;
                    idx = i;
                }
                // if a process has priority equal to max priority (min number) so far
                if (pr[i].priority == mn) {
                    // we chose the one that arrives first
                    if (pr[i].arrival < pr[idx].arrival) {
                        mn = pr[i].priority;
                        idx = i;
                    }
                }
            }
        }
        // if there exists a process
        if (idx != -1) {
            if (remaining[idx] == pr[idx].burst)
                pr[idx].start = current_time;
            remaining[idx] -= 1;
            current_time++;

            if (remaining[idx] == 0) {
                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++;
            }
        }
```

```
        else
            current_time++;

        // for Gantt chart
        g[cnt].idx = idx;
        g[cnt].start = current_time - 1;
        g[cnt].end = current_time;
        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 | Priority | 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              %2d\n", k[i].pid, k[i].priority, 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;
    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;
}

int main () {

    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    cout<<"\nPreemptive Priority 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
    int *p = new int[n];                    // priority

    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];
    cout<<"\nEnter the priority for each process: ";
    for (int i=0; i<n; i++)
        cin>>p[i];

    for (int i=0; i<n; i++) {
        // pr[i].pid = k[i];
        pr[i].pid = i+1;
        pr[i].arrival = at[i];
        pr[i].burst = bt[i];
        pr[i].priority = p[i];
        remaining[i] = pr[i].burst;
    }

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

    return 0;
}
```

OUTPUT:

```
FAIZAN CHOUDHARY
20BCS021

Preemptive Priority Scheduling Algorithm

Enter the number of processes: 6

Enter burst time for each process: 4 5 6 1 2 3

Enter arrival time for each process: 1 2 3 0 4 5

Enter the priority for each process: 5 2 6 4 7 8
```

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

   P1        5         4            1              14              9              13              9
   P2        2         5            2              12              5              10              5
   P3        6         6            3              22             13              19             13
   P4        4         1            0               2              1               2              1
   P5        7         2            4              20             14              16             14
   P6        8         3            5              24             16              19             16
------------------------------------------------------------------------------------------------------------------------


Average Completion time: 15.67
Average Waiting time: 9.67
Average Turnaround time: 13.17
Average Response time: 9.67

Gantt chart:
| P4 | P1 | P2 | P2 | P2 | P2 | P2 | P1 | P1 | P1 | P3 | P3 | P3 | P3 | P3 | P3 | P5 | P5 | P6 | P6 | P6 |
0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21
```

```
FAIZAN CHOUDHARY
20BCS021

Preemptive Priority Scheduling Algorithm

Enter the number of processes: 7

Enter burst time for each process: 4 2 3 5 1 4 6

Enter arrival time for each process: 0 1 2 3 4 5 6

Enter the priority for each process: 2 4 6 10 8 12 9
```

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

   P1        2         4            0               8              4               8              4
   P2        4         2            1               8              5               7              5
   P3        6         3            2              12              7              10              7
   P4       10         5            3              26             18              23             18
   P5        8         1            4              11              6               7              6
   P6       12         4            5              29             20              24             20
   P7        9         6            6              22             10              16             10
------------------------------------------------------------------------------------------------------------------------


Average Completion time: 16.57
Average Waiting time: 10.00
Average Turnaround time: 13.57
Average Response time: 10.00

Gantt chart:
| P1 | P1 | P1 | P1 | P2 | P2 | P3 | P3 | P3 | P5 | P7 | P7 | P7 | P7 | P7 | P7 | P4 | P4 | P4 | P4 | P4 | P6 | P6 | P6 | P6 |
0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25
```