Question :

Write a program to implement the Shortest Remaining Time First (Shortest job first preemptive) scheduling algorithm and find the average turnaround time, waiting time, completion time and response time for overall process. Also Print Gantt chart for it.

Garv nanwani

19BCS049

Code :

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

struct process {
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    int response_time;
};

int main() {

    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    int burst_remaining[100];
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));

    cout << setprecision(2) << fixed;

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

    for(int i = 0; i < n; i++) {
        cout<<"Enter arrival time of process "<<i+1<<": ";
        cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": ";
```

```
        cin>>p[i].burst_time;
        p[i].pid = i+1;
        burst_remaining[i] = p[i].burst_time;
        cout<<endl;
    }

    int current_time = 0;
    int completed = 0;
    int prev = 0;

    while(completed != n) {
        int idx = -1;
        int mn = 10000000;
        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
                if(burst_remaining[i] < mn) {
                    mn = burst_remaining[i];
                    idx = i;
                }
                if(burst_remaining[i] == mn) {
                    if(p[i].arrival_time < p[idx].arrival_time) {
                        mn = burst_remaining[i];
                        idx = i;
                    }
                }
            }
        }

        if(idx != -1) {
            if(burst_remaining[idx] == p[idx].burst_time) {
                p[idx].start_time = current_time;
                total_idle_time += p[idx].start_time - prev;
            }
            burst_remaining[idx] -= 1;
            current_time++;
            prev = current_time;

            if(burst_remaining[idx] == 0) {
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
                p[idx].response_time = p[idx].start_time - p[idx].arrival_time;

                total_turnaround_time += p[idx].turnaround_time;
                total_waiting_time += p[idx].waiting_time;
                total_response_time += p[idx].response_time;

                is_completed[idx] = 1;
                completed++;
            }
        }
        else {
```

```
            current_time++;
        }
    }

    int min_arrival_time = 10000000;
    int max_completion_time = -1;
    for(int i = 0; i < n; i++) {
        min_arrival_time = min(min_arrival_time,p[i].arrival_time);
        max_completion_time = max(max_completion_time,p[i].completion_time);
    }

    avg_turnaround_time = (float) total_turnaround_time / n;
    avg_waiting_time = (float) total_waiting_time / n;

    cout<<endl<<endl;

    cout<<"#P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"RT\t"<<"\n"<<endl;

    for(int i = 0; i < n; i++) {
        cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"
<<p[i].start_time<<"\t"<<p[i].completion_time<<"\t"<<p[i].turnaround_time<<"\t"
<<p[i].waiting_time<<"\t"<<p[i].response_time<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;


}
```

Output :

```
Enter the number of processes: 5
Enter arrival time of process 1: 3
Enter burst time of process 1: 6

Enter arrival time of process 2: 5
Enter burst time of process 2: 7

Enter arrival time of process 3: 3
Enter burst time of process 3: 6

Enter arrival time of process 4: 8
Enter burst time of process 4: 2

Enter arrival time of process 5: 5
Enter burst time of process 5: 7
```

```
#P      AT      BT      ST      CT      TAT     WT      RT

1       3       6       3       9       6       0       0

2       5       7       17      24      19      12      12

3       3       6       11      17      14      8       8

4       8       2       9       11      3       1       1

5       5       7       24      31      26      19      19

Average Turnaround Time = 13.60
Average Waiting Time = 8.00
```