CS212: Assignment 5

Md Shabbir Jamal

Department of Computer Science and Engineering
BIT, Mesra, Ranchi
btech10026.20@bitmesra.ac.in

1. WAP to schedule process according to Round Robin scheduling algorithm

```
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <queue>
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 time_q;
    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    float cpu_utilisation;
    int total_turnaround_time = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    float throughput;
    int burst_remaining[100];
    int idx;

bool a_t_sort(Process p1, Process p2)
{
    return p1.arrival_time < p2.arrival_time;
```

```
}

bool pid_sort(Process p1, Process p2)
{
    return p1.pid < p2.pid;
}

void input_process(vector<Process> &Proc)
{
    cout<<"Enter time quantum: ";
    cin>>time_q;

    for(int i = 0; i < Proc.size(); i++)
    {
        cout<<"p["<<i+1<<"] : ";

        cout<<"Arrival time : ";
        cin>>Proc[i].arrival_time;
        cout<<"       ";

        cout<<"Burst Time : ";
        cin>>Proc[i].burst_time;
        cout<<"       ";
        burst_remaining[i] = Proc[i].burst_time;
        Proc[i].pid = i+1;
        cout<<endl;
    }
}

void Gantt_Chart_n_Result(vector<Process> &Proc)
{
    cout << setprecision(2) << fixed;

    sort(Proc.begin(),Proc.end(),a_t_sort);

    queue<int> q;
    int current_time = 0;
    q.push(0);
    int completed = 0;
    vector<int> mark(Proc.size(),0);
    mark[0] = 1;

    while(completed != Proc.size())
    {
        idx = q.front();
        q.pop();

        if(burst_remaining[idx] == Proc[idx].burst_time)
```

```
{
    Proc[idx].start_time = max(current_time,Proc[idx].arrival_time);
    current_time = Proc[idx].start_time;
}

if(burst_remaining[idx]-time_q > 0)
{
    burst_remaining[idx] -= time_q;
    current_time += time_q;
}
else
{
    current_time += burst_remaining[idx];
    burst_remaining[idx] = 0;
    completed++;

    Proc[idx].completion_time = current_time;
    Proc[idx].turnaround_time = Proc[idx].completion_time - Proc[idx].arrival_time;
    Proc[idx].waiting_time = Proc[idx].turnaround_time - Proc[idx].burst_time;
    Proc[idx].response_time = Proc[idx].start_time - Proc[idx].arrival_time;

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

for(int i = 1; i < Proc.size(); i++)
{
    if(burst_remaining[i] > 0 && Proc[i].arrival_time <= current_time && mark[i] == 0) {
        q.push(i);
        mark[i] = 1;
    }
}
if(burst_remaining[idx] > 0) {
    q.push(idx);
}

if(q.empty()) {
    for(int i = 1; i < Proc.size(); i++) {
        if(burst_remaining[i] > 0) {
            q.push(i);
            mark[i] = 1;
            break;
        }
    }
}
```

```
    }

    avg_turnaround_time = (float) total_turnaround_time / Proc.size();
    avg_waiting_time = (float) total_waiting_time / Proc.size();
    avg_response_time = (float) total_response_time / Proc.size();
    throughput = float(Proc.size()) / (Proc[Proc.size() - 1].completion_time - Proc[0].arrival_t

    sort(Proc.begin(),Proc.end(),pid_sort);
    cout<<"Result : "<<endl;
    cout<<"Average Turnaround Time : "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time : "<<avg_waiting_time<<endl;
    cout<<"Average Response Time : "<<avg_response_time<<endl;
    cout<<"Throughput : "<<throughput<<endl;
}

int main() {

    int n;
    cout<<"\t\tEnter Process Details : "<<"\n";
    cout<<"Enter the number of Process : ";
    cin>>n;
    vector<Process> Proc(n);

    //input process detail
    input_process(Proc);

    Gantt_Chart_n_Result(Proc);

    return 0;
}
```

Output

```
Enter Process Details :
Enter the number of Process : 5
Enter time quantum: 2
p[1] : Arrival time : 0
       Burst Time : 5

p[2] : Arrival time : 1
       Burst Time : 3

p[3] : Arrival time : 2
       Burst Time : 1

p[4] : Arrival time : 3
```

```
       Burst Time : 2

p[5] : Arrival time : 4
       Burst Time : 3

Result :
Average Turnaround Time : 8.60
Average Waiting Time : 5.80
Average Response Time : 2.40
Throughput : 0.36
```