CS212: Assignment 4

Md Shabbir Jamal

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

1. WAP to schedule process according to Priority scheduling algorithm

```
#include<iostream>
#include<algorithm>
#include<vector>
#include<string>
using namespace std;

//structure to store process detail
struct Process
{
    string name;
    int arrival_time;
    int burst_time;
    int priority;
};

//helpful in sorting the process by arrival time
bool a_t_sort(Process P,Process Q)
{
    return P.arrival_time < Q.arrival_time;
}

bool b_t_sort(Process P,Process Q)
{
    return P.burst_time < Q.burst_time;
}

bool priority_sort(Process P, Process Q)
{
    return P.priority < Q.priority;
}

void input_process(vector<Process> &Proc)
{
    //taking input
    for(int i = 0;i<Proc.size();i++)
    {
        cout<<"p["<<i+1<<"] : ";
        Proc[i].name = to_string(i+1);

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

        cout<<"Burst Time : ";
        cin>>Proc[i].burst_time;

        cout<<"Priority : ";
        cin>>Proc[i].priority;
        cout<<"\n";
    }
}

void Gantt_chart_n_Result(vector<Process> &Proc)
{
    //Gantt Chart
    sort(Proc.begin(),Proc.end(),a_t_sort);

    int ttime=0;
    int j;
    vector<int> tArray(Proc.size());
    if(Proc[0].arrival_time != 0)
    {
        ttime  = Proc[0].arrival_time;
    }
    for(int i=0;i<Proc.size();i++)
    {
        j=i;
        while(Proc[j].arrival_time <= ttime && j != Proc.size())
        {
            j++;
        }
        sort(Proc.begin()+i,Proc.begin()+j,priority_sort);
        tArray[i]=ttime;
        ttime += Proc[i].burst_time;
    }
    tArray[Proc.size()] = ttime;

    cout<<"\nGantt Chart : "<<"\n\n";
    if(Proc[0].arrival_time != 0)
    {
        cout<<"|||";
    }
    for (int i=0; i<Proc.size(); i++)
    {
        cout<<" |||P["<< Proc[i].name << "]|||";
    }
    cout<<"\n";
    if(Proc[0].arrival_time != 0)
    {
        cout<<"0  ";
    }
    for (int i=0; i < Proc.size()+1 ; i++)
    {
        cout << tArray[i] << "          ";
    }

    cout<<"\n\nResults : \n\n";

    //Waiting Time
    double waiting_time = 0.0;
    int i = 0;
    for (int i = 0;i<Proc.size();i++)
    {
        waiting_time = waiting_time + (tArray[i] - Proc[i].arrival_time);
    }
     cout<<"Average Waiting Time : "<<waiting_time/Proc.size()<<"\n";

    //Turnaround Time
    double turnaround_time = 0.0;

    for(int i = 0;i < Proc.size();i++)
    {
        turnaround_time = turnaround_time + tArray[i] - Proc[i].arrival_time + Proc[i].burst_tim
    }
   cout<<"Average Turaround Time : "<<turnaround_time/Proc.size()<<"\n";
}

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

    //vector to store processes
    vector<Process> Proc(n);

    //input process detail
    input_process(Proc);
    //Calculate Gantt Chart and results
    Gantt_chart_n_Result(Proc);

    return 0;
}
```

Output

```
Enter Process Details :
Enter the number of Process : 4
p[1] : Arrival time : 0
       Burst Time : 5
       Priority : 2

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

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

p[4] : Arrival time : 4
       Burst Time : 1
       Priority : 4


Gantt Chart :

 |||P[1]||| |||P[3]||| |||P[2]||| |||P[4]|||
0          5          8          11          12

Results :

Average Waiting Time : 4.25
Average Turaround Time : 7.25
```