# 20BCS402 MOHD ADIL Program 7| Pre-emptive priority scheduling

// Preemtive priority scheduling

// Handled edge cases + idle Time

#include <iostream>

#include <vector>

using namespace std;

struct Process

{

    char Pname[3];

    int prTY;

    int Times[6];

    pair<int, int> scope;

    int b;

};

struct Gantt

{

    int s;

    int e;

    string pname;

};

// ans vector

vector<Process> v;

vector<Gantt> vG;

vector<bool> visited;

int n, CurrTime = 0;

float avgc, avgw, avgt, avgr;

bool allVisited()

{

    // traverse the visited array and find a false, hence return it

    for (auto b : visited)

    {

        if (!b)

            return false;

    }

    return true;

}

void SRTF()

{

    while (!allVisited())

    {

        int min = 1000;

        int idx = -1;

        for (int i = 0; i < v.size(); i++)

        {

            if (v[i].Times[0] <= CurrTime && v[i].b != 0)

            {

                if (v[i].prTY < min)

                {

                    idx = i;

                    min = v[i].prTY;

                }

            }

        }

        if (idx != -1)

        {

            int t = CurrTime;

            v[idx].b--;

            if (v[idx].b == 0)

            {

                visited[idx] = true;

            }

            if (v[idx].scope.first == -1)

            {

                v[idx].scope.first = t;

            }

            v[idx].scope.second = t + 1;

            Gantt g;

            g.s = t;

            g.e = t + 1;

            g.pname = v[idx].Pname;

            vG.push\_back(g);

        }

        CurrTime++;

    }

}

void calculateTimes()

{

    float sumc = 0, sumw = 0, sumt = 0, sumr = 0;

    // calculating completion time

    for (auto &p : v)

    {

        p.Times[2] = p.scope.second;

        sumc += p.Times[2];

    }

    // calculating turn around time

    //  CT-AR

    for (auto &p : v)

    {

        p.Times[4] = p.Times[2] - p.Times[0];

        sumt += p.Times[4];

    }

    // calculating waiting time

    //  TAT-BT

    for (auto &p : v)

    {

        p.Times[3] = p.Times[2] - p.Times[0] - p.Times[1];

        sumw += p.Times[3];

    }

    // calculating Response Time

    //  First - AT

    for (auto &p : v)

    {

        p.Times[5] = p.scope.first - p.Times[0];

        sumr += p.Times[5];

    }

    // calculating avg(s) time

    avgc = sumc / n;

    avgw = sumw / n;

    avgt = sumt / n;

    avgr = sumr / n;

}

void display()

{

    cout << "\n\nDisplaying the table :- ";

    cout << "\n\n+--------------+----------+------------+--------------+-----------------+--------------+-----------------+---------------+";

    cout << "\n| Process name | Priority | Burst Time | Arrival Time | Completion Time | Waiting Time | TurnAround Time | Response Time |";

    cout << "\n+--------------+----------+------------+--------------+-----------------+--------------+-----------------+---------------+";

    // cout<<"\n+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+";

    for (auto i : v)

    {

        printf("\n|      %s      |    %2d    |    %2d      |      %2d      |        %2d       |      %2d      |      %2d         |      %2d       |", i.Pname, i.prTY, i.Times[1], i.Times[0], i.Times[2], i.Times[3], i.Times[4], i.Times[5]);

        // cout<<"\n+--------------+------------+--------------+-----------------+--------------+-----------------+---------------+";

        cout << "\n+--------------+----------+------------+--------------+-----------------+--------------+-----------------+---------------+";

    }

    cout << "\n\n";

    printf("\nAverage Completion time : %.2fms", avgc);

    printf("\nAverage Waiting time : %.2fms", avgw);

    printf("\nAverage TurnAround time : %.2fms", avgt);

    printf("\nAverage Response time : %.2fms", avgr);

}

void PrintGantt()

{

    cout << endl

         << endl

         << "Gantt Chart : " << endl

         << endl;

    cout << "--------------------------------------------------------------------------------------------------------------";

    cout << endl;

    vector<int> t;

    // vector<pair<int,int>> indices;

    string prv = "-1";

    for (int i = 0; i < CurrTime; i++)

    {

        string ch = "--";

        for (auto g : vG)

        {

            if (g.s == i)

            {

                ch = g.pname;

                break;

            }

        }

        if (prv != ch)

        {

            cout << "| " << ch << "  ";

            t.push\_back(i);

        }

        else

            cout << "    ";

        prv = ch;

    }

    cout << "|" << endl;

    t.push\_back(CurrTime);

    cout << "--------------------------------------------------------------------------------------------------------------";

    cout << endl;

    int prev = 0;

    for (int i = 0; i < t.size(); i++)

    {

        for (int j = 0; j < (t[i] - prev); j++)

        {

            cout << "    ";

        }

        // cout<<t[i];

        printf("%2d", t[i]);

        prev = t[i];

    }

}

int main()

{

    cout << "Enter the no of the Processes : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

        struct Process p;

        cout << "Enter Process " << i + 1 << " name, Priority, Arrival Time and Burst Time: ";

        cin >> p.Pname >> p.prTY >> p.Times[0] >> p.Times[1];

        p.b = p.Times[1];

        visited.push\_back(false);

        p.scope.first = -1;

        p.scope.second = -1;

        v.push\_back(p);

    }

    SRTF();

    calculateTimes();

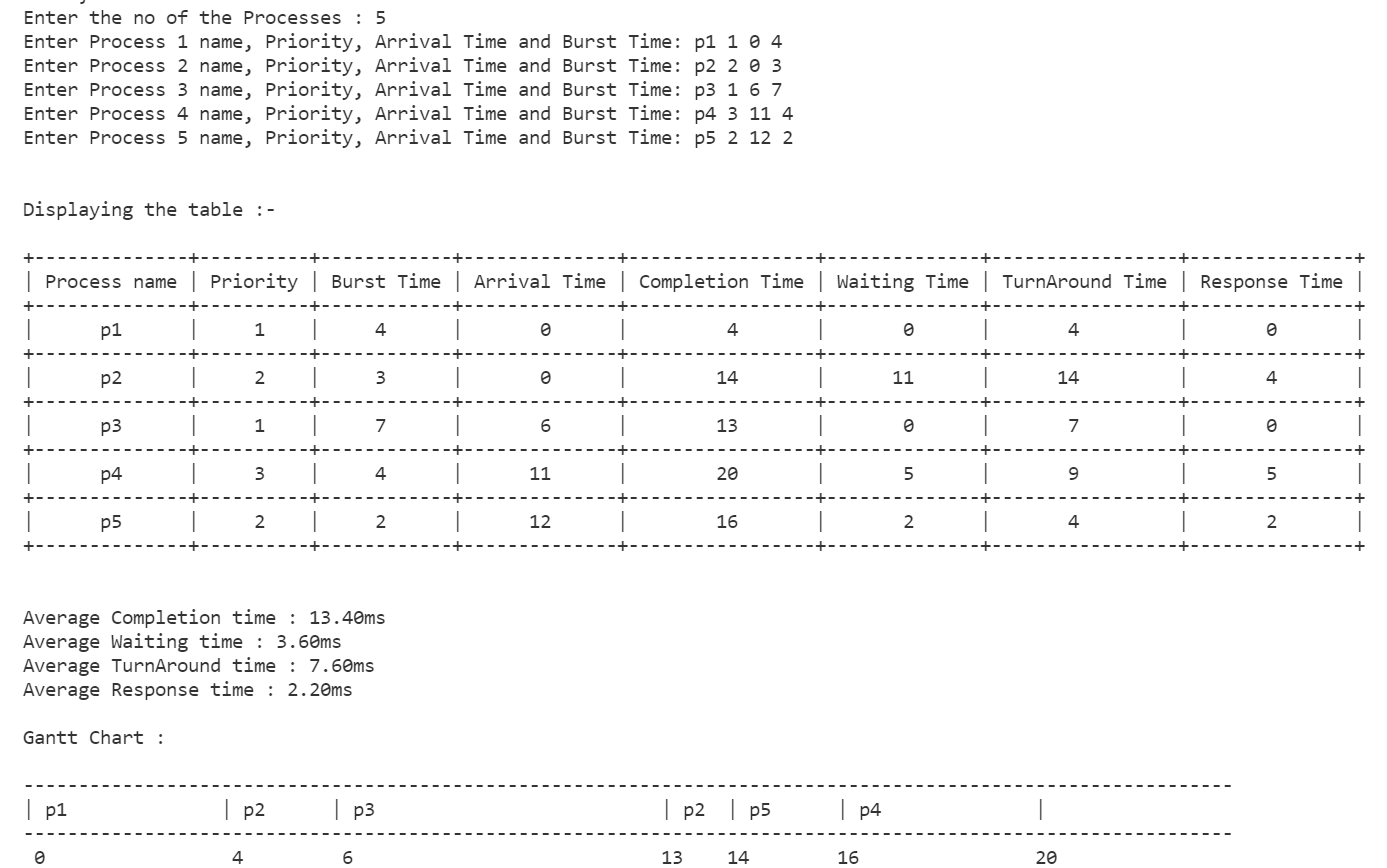
    display();

    PrintGantt();

    return 0;

}

**OUTPUT**

****

Thank you