\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operating System Lab

CEN-493

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program - 2

Code :-

#include <iostream>

using namespace std;

struct Process

{

    string pname;

    int arival\_time;

    int burst\_time;

    int waiting\_time;

    int completion\_time;

    int response\_time;

    int turnaound\_time;

};

void Print\_Bars()

{

    for (int i = 0; i < 100; i++)

        cout << "\_";

    cout << "\n";

}

void Insertion\_Sort(Process Process\_Array[], int total\_process)

{

    for (int i = 1; i < total\_process; i++)

    {

        Process curent = Process\_Array[i];

        int j = i - 1;

        while (Process\_Array[j].arival\_time > curent.arival\_time && j >= 0)

        {

            Process\_Array[j + 1] = Process\_Array[j];

            j--;

        }

        Process\_Array[j + 1] = curent;

    }

}

void Average\_Time(Process Process\_Array[], int total\_process)

{

    double Av\_CT = 0, Av\_RT = 0, Av\_WT = 0, Av\_TAT = 0;

    for (int i = 0; i < total\_process; i++)

    {

        Av\_CT += Process\_Array[i].completion\_time;

        Av\_RT += Process\_Array[i].response\_time;

        Av\_TAT += Process\_Array[i].turnaound\_time;

        Av\_WT += Process\_Array[i].waiting\_time;

    }

    Av\_WT /= total\_process;

    Av\_TAT /= total\_process;

    Av\_RT /= total\_process;

    Av\_CT /= total\_process;

    cout << "Average Time For The Different Time In Process Scheduling\n\n";

    cout << "Average Completion Time -> " << Av\_CT << "\n";

    cout << "Average Waiting Time -> " << Av\_WT << "\n";

    cout << "Average Turn Around Time -> " << Av\_TAT << "\n";

    cout << "Average Respond Time -> " << Av\_RT << "\n";

}

void GanttChart(Process Process\_Array[], int total\_process)

{

    cout << "Gantt Chart For Process Scheduling\n";

    cout << "\n";

    if (Process\_Array[0].arival\_time != 0)

    {

        cout << "|     |  ";

    }

    else

    {

        cout << "|  ";

    }

    for (int i = 0; i < total\_process; i++)

    {

        if (i != 0 && Process\_Array[i - 1].completion\_time < Process\_Array[i].arival\_time)

        {

            cout << "    |   ";

        }

        cout << Process\_Array[i].pname << "  |   ";

    }

    cout << "\n";

    if (Process\_Array[0].arival\_time != 0)

    {

        cout << " 0     ";

        cout << Process\_Array[0].arival\_time << "     ";

    }

    else

    {

        cout << Process\_Array[0].arival\_time << "      ";

    }

    for (int i = 0; i < total\_process; i++)

    {

        if (i != 0 && Process\_Array[i - 1].completion\_time < Process\_Array[i].arival\_time)

        {

            cout << Process\_Array[i].arival\_time << "      ";

        }

        cout << Process\_Array[i].completion\_time << "      ";

    }

    cout << "\n";

}

void Chart(Process Process\_Array[], int total\_process)

{

    cout << "Various Time's Related To Process Scheduling\n\n";

    cout << "|  Process   |  BT  |  AT  |   CT   |   WT  |  TAT  |  RT  |\n";

    for (int i = 0; i < total\_process; i++)

    {

        cout << "   " << Process\_Array[i].pname << "\t\t" << Process\_Array[i].burst\_time

             << "\t" << Process\_Array[i].arival\_time << "\t" << Process\_Array[i].completion\_time

             << "\t" << Process\_Array[i].waiting\_time << "\t" << Process\_Array[i].turnaound\_time

             << "\t" << Process\_Array[i].response\_time << "\n";

    }

}

void FCFS(Process Process\_Array[], int total\_process)

{

    Insertion\_Sort(Process\_Array, total\_process); // Acording To A.T

    int timer = 0;

    for (int i = 0; i < total\_process; i++)

    {

        if (timer < Process\_Array[i].arival\_time)

        {

            timer += (Process\_Array[i].arival\_time - timer);

        }

        timer += Process\_Array[i].burst\_time;

        Process\_Array[i].completion\_time = timer;

        Process\_Array[i].turnaound\_time =

            Process\_Array[i].completion\_time -

            Process\_Array[i].arival\_time;

        Process\_Array[i].waiting\_time =

            Process\_Array[i].turnaound\_time -

            Process\_Array[i].burst\_time;

        Process\_Array[i].response\_time = Process\_Array[i].waiting\_time;

    }

    Print\_Bars();

    Chart(Process\_Array, total\_process);

    Print\_Bars();

    Print\_Bars();

    GanttChart(Process\_Array, total\_process);

    Print\_Bars();

    Print\_Bars();

    Average\_Time(Process\_Array, total\_process);

    Print\_Bars();

}

int main()

{

    system("cls");

    Print\_Bars();

    cout << "20BCS070\_Vicky\_Gupta\n";

    cout << "First Come First Serve Process Scheduling Alogorithm\n";

    Print\_Bars();

    int total\_process;

    cout << "Enter The No Of Processes : ";

    cin >> total\_process;

    fflush(stdin);

    Process Process\_Array[total\_process];

    Print\_Bars();

    cout << "Enter The Process Details...\n";

    cout << "| Process Name | Burst Time | Arival Time | \n";

    for (int i = 0; i < total\_process; i++)

    {

        cin >> Process\_Array[i].pname;

        cin >> Process\_Array[i].burst\_time;

        cin >> Process\_Array[i].arival\_time;

    }

    FCFS(Process\_Array, total\_process);

    Print\_Bars();

    cout << "Exited..\n";

    Print\_Bars();

    return 0;

}

Output :-

Text

Description automatically generated