Operating System Lab

CEN-493

Program - 3

Code :-

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

struct Process
{
    string P_Name;
    int AT;
    int BT;
    int WT;
    int CT;
    int RT;
    int TAT;
};

bool mycomp(Process P1, Process P2)
```

```
{

    if (P1.AT != P2.AT)
    {
        return P1.AT < P2.AT;
    }
    else if (P1.BT != P2.BT)
    {
        return P1.BT < P2.BT;
    }
    else
    {
        int num1 = stoi(P1.P_Name.substr(1));
        int num2 = stoi(P2.P_Name.substr(1));
        return num1 < num2;
    }
}

void Print_Bars()
{
    for (int i = 0; i < 100; i++)
        cout << "_";
    cout << "\n";
}

void Average_Time(Process P_Array[], int T_Process)
{
    double Av_CT = 0, Av_RT = 0, Av_WT = 0, Av_TAT = 0;
    for (int i = 0; i < T_Process; i++)
    {
        Av_CT += P_Array[i].CT;
        Av_RT += P_Array[i].RT;
        Av_TAT += P_Array[i].TAT;
        Av_WT += P_Array[i].WT;
    }
    Av_WT /= T_Process;
    Av_TAT /= T_Process;
    Av_RT /= T_Process;
    Av_CT /= T_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 P_Array[], int T_Process)
{

    cout << "Gantt Chart For Process Scheduling\n";
    cout << "\n";
    if (P_Array[0].AT != 0)
    {
        cout << "|     |  ";
    }
    else
    {
        cout << "|  ";
    }

    for (int i = 0; i < T_Process; i++)
    {
        if (i != 0 && P_Array[i - 1].CT < P_Array[i].AT)
        {
            cout << "    |   ";
        }
        cout << P_Array[i].P_Name << "  |   ";
    }
    cout << "\n";

    if (P_Array[0].AT != 0)
    {
        cout << " 0     ";
        cout << P_Array[0].AT << "     ";
```

```
    }
    else
    {
        cout << P_Array[0].AT << "      ";
    }

    for (int i = 0; i < T_Process; i++)
    {
        if (i != 0 && P_Array[i - 1].CT < P_Array[i].AT)
        {
            cout << P_Array[i].AT << "      ";
        }
        cout << P_Array[i].CT << "      ";
    }
    cout << "\n";
}

void Chart(Process P_Array[], int T_Process)
{
    cout << "Various Time's Related To Process
Scheduling\n\n";
    cout <<
"|  Process   |  AT  |  BT  |   CT   |   WT  |  TAT  |  R
T  |\n";
    for (int i = 0; i < T_Process; i++)
    {
        cout << "    " << P_Array[i].P_Name << "\t\t" <<
P_Array[i].AT
             << "\t" << P_Array[i].BT << "\t" <<
P_Array[i].CT
             << "\t" << P_Array[i].WT << "\t" <<
P_Array[i].TAT
             << "\t" << P_Array[i].RT << "\n";
    }
}

void New_Process_Array(Process P_Array[], Process
N_P_Array[], int T_Process)
{
```

```
    sort(P_Array, P_Array + T_Process, mycomp);
    bool isProcessed[T_Process] = {0};
    int Timer = P_Array[0].AT;
    for (int i = 0; i < T_Process; i++)
    {
        int p_no = -1;
        for (int j = 0; j < T_Process; j++)
        {
            if (Timer >= P_Array[j].AT && isProcessed[j]
== 0)
            {
                if (p_no == -1)
                {
                    p_no = j;
                }
                if (p_no != -1 && P_Array[p_no].BT >
P_Array[j].BT)
                {
                    p_no = j;
                }
            }
        }
        if (p_no == -1) // when the process has gaps
        {
            for (int j = 0; j < T_Process; j++)
            {
                if (isProcessed[j] == 0)
                {
                    p_no = j;
                    break;
                }
            }
        }
        isProcessed[p_no] = 1;
        N_P_Array[i] = P_Array[p_no];
        if (Timer < P_Array[p_no].AT)
        {
            Timer += (P_Array[p_no].AT - Timer);
        }
```

```
            Timer += P_Array[p_no].BT;
        }
    }

    void SJF(Process P_Array[], int T_Process)
    {
        Process N_P_Array[T_Process];
        New_Process_Array(P_Array, N_P_Array, T_Process);

        int Timer = 0;
        for (int i = 0; i < T_Process; i++)
        {

            if (Timer < N_P_Array[i].AT)
            {
                Timer += (N_P_Array[i].AT - Timer);
            }
            Timer += N_P_Array[i].BT;

            N_P_Array[i].CT = Timer;

            N_P_Array[i].TAT = N_P_Array[i].CT -
    N_P_Array[i].AT;

            N_P_Array[i].WT = N_P_Array[i].TAT -
    N_P_Array[i].BT;

            N_P_Array[i].RT = N_P_Array[i].WT;
        }
        Print_Bars();
        Chart(N_P_Array, T_Process);
        Print_Bars();
        Print_Bars();
        GanttChart(N_P_Array, T_Process);
        Print_Bars();
        Print_Bars();
        Average_Time(N_P_Array, T_Process);
        Print_Bars();
    }
```

```
int main()
{
    // system("cls");
    Print_Bars();
    cout << "20BCS070_Vicky_Gupta\n";
    cout << "Shortest Job First Process Scheduling
Alogorithm\n";
    Print_Bars();
    int T_Process;
    cout << "Enter The No Of Processes : ";
    cin >> T_Process;
    fflush(stdin);
    Process P_Array[T_Process];
    Print_Bars();
    cout << "Enter The Process Details...\n";
    cout << "| Process Name | Arival Time | Burst Time |
\n";

    for (int i = 0; i < T_Process; i++)
    {
        cin >> P_Array[i].P_Name;
        cin >> P_Array[i].AT;
        cin >> P_Array[i].BT;
    }

    SJF(P_Array, T_Process);
    Print_Bars();
    cout << "Exited..\n";
    Print_Bars();
    return 0;
}
```

Output :-

```
-------------------------------------------------------------------
20BCS070_Vicky_Gupta
Shortest Job First Process Scheduling Alogorithm
-------------------------------------------------------------------
Enter The No Of Processes : 4
-------------------------------------------------------------------
Enter The Process Details...
| Process Name | Arival Time | Burst Time |
P1 2 4
P2 6 5
P3 6 5
P4 40 1
-------------------------------------------------------------------
Various Time's Related To Process Scheduling

|  Process   |  AT  |  BT  |   CT   |   WT  |  TAT  |  RT  |
   P1           2      4       6       0       4       0
   P2           6      5       11      0       5       0
   P3           6      5       16      5       10      5
   P4           40     1       41      0       1       0
-------------------------------------------------------------------
-------------------------------------------------------------------
Gantt Chart For Process Scheduling

|     |  P1  |   P2  |   P3  |       |   P4  |
 0     2     6      11      16      40      41
-------------------------------------------------------------------
-------------------------------------------------------------------
Average Time For The Different Time In Process Scheduling

Average Completion Time -> 18.5
Average Waiting Time -> 1.25
Average Turn Around Time -> 5
Average Respond Time -> 1.25
-------------------------------------------------------------------
-------------------------------------------------------------------
Exited..
-------------------------------------------------------------------
```