
Operating System Lab

CEN-493

Program - 5

Code :-

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
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
    {
        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 < 120; 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(vector<pair<string, pair<int, int>>>
&All_Interval)
{
    int size = All_Interval.size();
    cout << "Gantt Chart For Process Scheduling\n";
    cout << "\n";
    if (All_Interval[0].second.first != 0)
    {
        cout << "| \t \t |  ";
    }
    else
    {
        cout << "| \t ";
    }

    for (int i = 0; i < size; i++)
    {
        if (i != 0 && All_Interval[i - 1].second.second <
All_Interval[i].second.first)
        {
            cout << " \t | \t ";
        }
        cout << All_Interval[i].first << " \t | \t ";
    }
    cout << "\n";

    if (All_Interval[0].second.first != 0)
    {
        cout << " 0 \t ";
        cout << All_Interval[0].second.first << " \t ";
    }
}

```

```

    }
    else
    {
        cout << All_Interval[0].second.first << "\t\t";
    }

    for (int i = 0; i < size; i++)
    {
        if (i != 0 && All_Interval[i - 1].second.second <
All_Interval[i].second.first)
        {
            cout << All_Interval[i].second.first <<
"\t\t";
        }
        cout << All_Interval[i].second.second << "\t\t";
    }
    cout << "\n";
}

void Chart(Process P_Array[], int T_Process)
{
    cout << "Various Time's Related To Process
Scheduling\n\n";
    cout << "+-----+
-----+
-----+\n";
    cout <<
"|\tProcess\t|\tAT\t|\tBT\t|\tCT\t|\tWT\t|\tTAT\t|\tRT
|\n";
    cout << "+-----+
-----+
-----+\n";
    for (int i = 0; i < T_Process; i++)
    {
        cout << "|\t" << P_Array[i].P_Name << "\t|\t" <<
P_Array[i].AT
        << "\t|\t" << P_Array[i].BT << "\t|\t" <<
P_Array[i].CT

```



```

    Chart(P_Array, T_Process);
    Print_Bars();
    Average_Time(P_Array, T_Process);
    Print_Bars();
    GanttChart(All_Interval);
    Print_Bars();
}

vector<pair<string, pair<int, int>>>
Time_Intervals(vector<string> &timeArray)
{
    vector<pair<string, pair<int, int>>>
processTimeInterval;
    for (int i = 0; i < timeArray.size(); i++)
    {
        int end = timeArray.size();
        for (int j = i + 1; j < timeArray.size(); j++)
        {
            if (timeArray[i] != timeArray[j])
            {
                end = j;
                break;
            }
        }
        processTimeInterval.push_back({timeArray[i], {i,
end}}});
        i = end - 1;
    }
    return processTimeInterval;
}

void AddTimeToArray(Process process, vector<string>
&timeArray, int timer, int TQ)
{
    for (int i = timer; i < timer + TQ; i++)
    {
        timeArray.push_back(process.P_Name);
    }
}

```

```

void RoundRobin_Preemptive(Process P_Array[], int
T_Process, int TQ)
{
    sort(P_Array, P_Array + T_Process, mycomp);
    queue<Process> que;
    int processIterator = 0;
    vector<string> timeArray;
    que.push(P_Array[0]);
    int timer = P_Array[processIterator].AT;
    if (timer != 0)
    {
        Process pnull;
        pnull.P_Name = "--";
        AddTimeToArray(pnull, timeArray, 0, timer);
    }
    processIterator++;
    while (!que.empty() || processIterator < T_Process)
    {
        if (!que.empty())
        {
            Process processCpuAllocated = que.front();
            que.pop();
            while (processIterator < T_Process && timer +
min(TQ, processCpuAllocated.BT) >=
P_Array[processIterator].AT)
            {
                que.push(P_Array[processIterator++]);
            }
            if (processCpuAllocated.BT > TQ)
            {
                processCpuAllocated.BT -= TQ;
                AddTimeToArray(processCpuAllocated,
timeArray, timer, TQ);
                que.push(processCpuAllocated);
                timer += TQ;
            }
            else
            {

```

```

        int remTime = processCpuAllocated.BT;
        AddTimeToArray(processCpuAllocated,
timeArray, timer, remTime);
        timer += remTime;
    }
}
else
{
    timeArray.push_back("--");
    timer++;
    while (processIterator < T_Process && timer
>= P_Array[processIterator].AT)
    {
        que.push(P_Array[processIterator++]);
    }
}
}
vector<pair<string, pair<int, int>>> Intervals =
Time_Intervals(timeArray);
Timing(Intervals, P_Array, T_Process);
}

```

```

int main()
{
    system("cls");
    Print_Bars();
    cout << "20BCS070_Vicky_Gupta\n";
    cout << "Round Robin Process Scheduling
Algorithm\n";
    Print_Bars();
    int T_Process;
    cout << "Enter The No Of Processes : ";
    cin >> T_Process;
    int TQ;
    cout << "Enter The Time Quantum : ";
    cin >> TQ;
    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;
    }

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

Output :-

20BCS070_Vicky_Gupta

Round Robin Process Scheduling Alogorithm

Enter The No Of Processes : 4

Enter The Time Quantum : 2

Enter The Process Details...

| Process Name | Arival Time | Burst Time |

P1 1 4

P2 2 1

P3 3 8

P4 4 1

Various Time's Related To Process Scheduling

Process	AT	BT	CT	WT	TAT	RT
P1	1	4	8	3	7	0
P2	2	1	4	1	2	1
P3	3	8	15	4	12	1
P4	4	1	9	4	5	4

Average Time For The Different Time In Process Scheduling

Average Completion Time -> 9

Average Waiting Time -> 3

Average Turn Around Time -> 6.5

Average Respond Time -> 1.5

Gantt Chart For Process Scheduling

	--		P1		P2		P3		P1		P4		P3	
0		1		3		4		6		8		9		15

Exited..