Operating System Lab

CEN-493

Program - 4

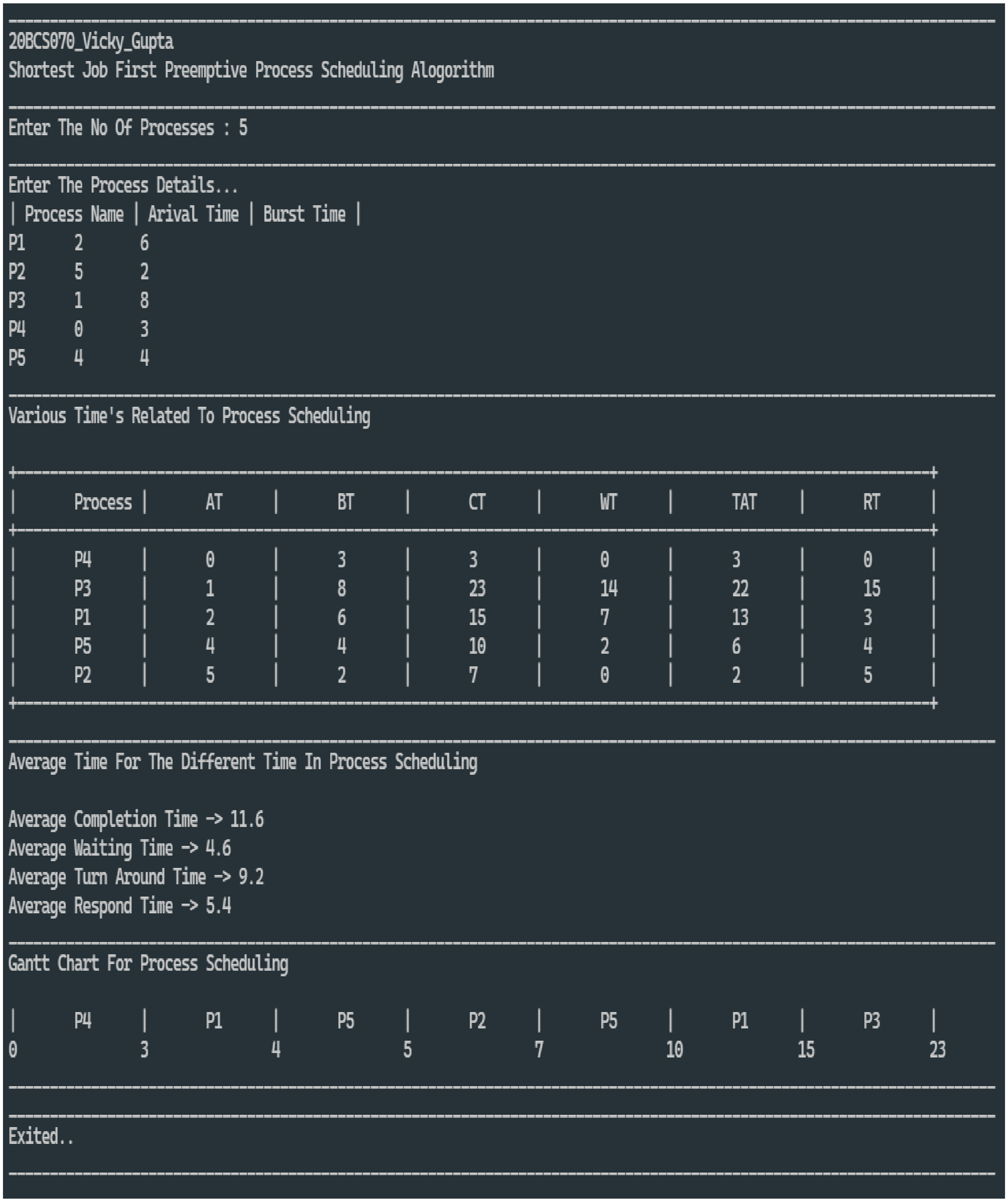
Code :-

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <unordered_map>
#include <stack>
using namespace std;

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

```cpp
struct myCompBT
{
    bool operator()(Process &p1, Process const &p2)
    {
        return p1.BT > p2.BT;
    }
};

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;
    }
}

bool mycompInterval(pair<string, pair<int, int>> p1, pair<string,
pair<int, int>> p2)
{
    return p1.second.first < p2.second.first;
}

vector<pair<string, pair<int, int>>>
Merge_Interval_Helper(vector<pair<int, int>> Interval, string
P_Name)
{

    stack<pair<int, int>> helper;
    int Interval_Length = Interval.size();
    helper.push(Interval[0]);
    for (int i = 1; i < Interval_Length; i++)
    {
        if (Interval[i].first <= helper.top().second)
        {
            helper.top().second = Interval[i].second;
        }
        else
        {
            helper.push(Interval[i]);
        }
    }
    vector<pair<string, pair<int, int>>> result;
    while (!helper.empty())
    {
        result.push_back({P_Name, {helper.top().first,
helper.top().second}});
        helper.pop();
    }
    return result;
}

vector<pair<string, pair<int, int>>>
Merge_Interval(unordered_map<string, vector<pair<int, int>>>
&executionTime)
{
    vector<pair<string, pair<int, int>>> Intervals;
    for (auto &x : executionTime)
    {
        vector<pair<string, pair<int, int>>> intvl =
Merge_Interval_Helper(x.second, x.first);
        for (auto &y : intvl)
        {
            Intervals.push_back(y);
        }
    }
    sort(Intervals.begin(), Intervals.end(), mycompInterval);
    return Intervals;
}

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
             << "\t|\t" << P_Array[i].WT << "\t|\t" <<
P_Array[i].TAT
             << "\t|\t" << P_Array[i].RT << "\t|\n";
    }
    cout << "+----------------------------------------------------
-----------------------------------------------------------+\n";
}
```

```
void Timing(vector<pair<string, pair<int, int>>> &All_Interval,
Process P_Array[], int T_Process)
{
    int size = All_Interval.size();
    for (int i = 0; i < T_Process; i++)
    {
        for (int j = size - 1; j >= 0; j--)
        {
            if (P_Array[i].P_Name == All_Interval[j].first)
            {
                P_Array[i].CT = All_Interval[j].second.second;
                break;
            }
        }
        P_Array[i].TAT = P_Array[i].CT - P_Array[i].AT;
        P_Array[i].WT = P_Array[i].TAT - P_Array[i].BT;
        for (int j = 0; j < size; j++)
        {
            if (P_Array[i].P_Name == All_Interval[j].first)
            {
                P_Array[i].RT = All_Interval[j].second.first;
                break;
            }
        }
    }
    Print_Bars();
    Chart(P_Array, T_Process);
    Print_Bars();
    Average_Time(P_Array, T_Process);
    Print_Bars();
    GanttChart(All_Interval);
    Print_Bars();
}

void SJF_Preemptive(Process P_Array[], int T_Process)
{
    sort(P_Array, P_Array + T_Process, mycomp);
    priority_queue<Process, vector<Process>, myCompBT> pque;
    unordered_map<string, vector<pair<int, int>>> executionTime;
    int processItertor = 0;
    int timer = P_Array[processItertor].AT;
    pque.push(P_Array[processItertor]);
    if (timer != 0)
    {
        executionTime[P_Array[processItertor].P_Name].push_back({0
, timer});
    }
    processItertor++;
    while (!pque.empty() || processItertor < T_Process)
    {
        timer++;
        if (!pque.empty())
        {
            Process process = pque.top();
            pque.pop();
            process.BT--;
            executionTime[process.P_Name].push_back({timer - 1,
timer});
            if (process.BT != 0)
                pque.push(process);
        }
        while (processItertor < T_Process && timer >=
P_Array[processItertor].AT)
        {
            pque.push(P_Array[processItertor++]);
        }
    }

    vector<pair<string, pair<int, int>>> All_Interval =
Merge_Interval(executionTime);
    Timing(All_Interval, P_Array, T_Process);
}

int main()
{
    system("cls");
    Print_Bars();
    cout << "20BCS070_Vicky_Gupta\n";
    cout << "Shortest Job First Preemptive 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_Preemptive(P_Array, T_Process);
    Print_Bars();
    cout << "Exited..\n";
    Print_Bars();
    return 0;
}
```

Output :-



```
-------------------------------------------------------------------------------------------------------------
20BCS070_Vicky_Gupta
Shortest Job First Preemptive Process Scheduling Alogorithm

-------------------------------------------------------------------------------------------------------------
Enter The No Of Processes : 5

-------------------------------------------------------------------------------------------------------------
Enter The Process Details...
| Process Name | Arival Time | Burst Time |
P1      2       6
P2      5       2
P3      1       8
P4      0       3
P5      4       4

-------------------------------------------------------------------------------------------------------------
Various Time's Related To Process Scheduling


+------------------------------------------------------------------------------------------------------+
|       Process |       AT      |       BT      |       CT      |       WT      |       TAT     |       RT      |
+------------------------------------------------------------------------------------------------------+
|       P4      |       0       |       3       |       3       |       0       |       3       |       0       |
|       P3      |       1       |       8       |       23      |       14      |       22      |       15      |
|       P1      |       2       |       6       |       15      |       7       |       13      |       3       |
|       P5      |       4       |       4       |       10      |       2       |       6       |       4       |
|       P2      |       5       |       2       |       7       |       0       |       2       |       5       |
+------------------------------------------------------------------------------------------------------+

-------------------------------------------------------------------------------------------------------------
Average Time For The Different Time In Process Scheduling

Average Completion Time -> 11.6
Average Waiting Time -> 4.6
Average Turn Around Time -> 9.2
Average Respond Time -> 5.4

-------------------------------------------------------------------------------------------------------------
Gantt Chart For Process Scheduling

|       P4      |       P1      |       P5      |       P2      |       P5      |       P1      |       P3      |
0               3               4               5               7               10              15              23
-------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------
Exited..

-------------------------------------------------------------------------------------------------------------
```