RALF A. CAGBAY
BSCS 3-1

DEC 15, 2025

I. CODE

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    int at[n], bt[n], pr[n];
    int rt[n], ct[n], tat[n], wt[n];

    for (int i = 0; i < n; i++) {
        cout << "\nProcess P" << i + 1 << endl;
        cout << "Arrival Time: ";
        cin >> at[i];
        cout << "Burst Time: ";
        cin >> bt[i];
        cout << "Priority (lower = higher): ";
        cin >> pr[i];
        rt[i] = bt[i];
    }

    int time = 0, completed = 0, idleTime = 0;
    int prev = -1;

    // Gantt storage
    int gProc[100], gTime[100], gCount = 0;

    while (completed < n) {
        int idx = -1, minPr = 9999;

        for (int i = 0; i < n; i++) {
            if (at[i] <= time && rt[i] > 0 && pr[i] < minPr) {
                minPr = pr[i];
                idx = i;
            }
        }

        // CPU Idle
        if (idx == -1) {
            if (prev != -2) {
                gProc[gCount] = -1;
                gTime[gCount++] = time;
                prev = -2;
            }
            idleTime++;
            time++;
            continue;
        }

        // Process change
        if (prev != idx) {
            gProc[gCount] = idx;
            gTime[gCount++] = time;
            prev = idx;
        }

        rt[idx]--;
        time++;

        if (rt[idx] == 0) {
            ct[idx] = time;
            tat[idx] = ct[idx] - at[idx];
            wt[idx] = tat[idx] - bt[idx];
            completed++;
        }
    }

    gTime[gCount] = time;
```

```cpp
    // Print Gantt Chart
    cout << "\nGANTT CHART\n|";
    for (int i = 0; i < gCount; i++) {
        if (gProc[i] == -1)
            cout << " IDLE |";
        else
            cout << " P" << gProc[i] + 1 << " |";
    }

    cout << "\n";
    for (int i = 0; i <= gCount; i++) {
        cout << setw(5) << gTime[i];
    }

    // Averages
    float avgWT = 0, avgTAT = 0;
    for (int i = 0; i < n; i++) {
        avgWT += wt[i];
        avgTAT += tat[i];
    }

    avgWT /= n;
    avgTAT /= n;

    float cpuUtil = ((float)(time - idleTime) / time) * 100;

    // Table
    cout << "\n\nPROCESS DETAILS\n";
    cout << "P\tAT\tBT\tPR\tCT\tTAT\tWT\n";

    for (int i = 0; i < n; i++) {
        cout << "P" << i + 1 << "\t"
             << at[i] << "\t"
             << bt[i] << "\t"
             << pr[i] << "\t"
             << ct[i] << "\t"
             << tat[i] << "\t"
             << wt[i] << endl;
    }

    cout << fixed << setprecision(2);
    cout << "\nAverage TAT = " << avgTAT;
    cout << "\nAverage WT  = " << avgWT;
    cout << "\nCPU Utilization = " << cpuUtil << "%\n";

    return 0;
}
```

II. OUTPUT AND GANTT CHART

```
Enter number of processes: 4

Process P1
Arrival Time: 0
Burst Time: 7
Priority (lower = higher): 3

Process P2
Arrival Time: 2
Burst Time: 4
Priority (lower = higher): 1

Process P3
Arrival Time: 4
Burst Time: 3
Priority (lower = higher): 2

Process P4
Arrival Time: 6
Burst Time: 2
Priority (lower = higher): 0
```

```
PROCESS DETAILS
P       AT      BT      PR      CT      TAT     WT
P1      0       7       3       16      16      9
P2      2       4       1       6       4       0
P3      4       3       2       11      7       4
P4      6       2       0       8       2       0
```

```
Average TAT = 7.25
Average WT  = 3.25
CPU Utilization = 100.00%
```

```
| P1 | P2 | P4 | P3 | P1 |
0    2    6    8    11   16
```

III. STEP BY STEP

TIME	AVAILABLE PROCESS	EXECUTION
0	P1	P1 is the only available process, so it will execute.
1	-----	P1 burst time = 6. Still processing.
2	P2	P1 burst time = 5. Since this is priority preemptive, the priority of P1 and P2 is compared. Because P2 has a higher priority, P2 will execute next.
3	P1	P2 BT = 3. STILL PROCESSING
4	P1,P3	P2 burst time = 2. The system checks the priority of all available processes. Since P2 still has the highest priority, it continues executing.
5	P1,P3	P2 BT = 1. STILL PROCESSING
6	P1,P3,P4	P2 has finished. Now the priorities of all available processes are compared. Since P4 has the highest priority, P4 will execute.
7	P1,P3	P4 BT = 1, STILL PROCESSING
8	P1,P3	P4 has finished. The remaining processes are compared. Since P3 has a higher priority than P1, P3 will execute first.
9	P1	P3 BT = 2
10	P1	P3 BT = 1
11	P1	P3 has finished. Since P1 is the only available process, it will continue executing until completion.
16	-----	P1 has finished. All processes are completed.

COMPUTATION:

Process	CT – AT = TAT	TAT – BT = WT
P1	16 − 0 = **16**	16 − 7 = **9**
P2	6 − 2 = **4**	4 − 4 = **0**
P3	11 − 4 = **7**	7 − 3 = **4**
P4	8 − 6 = **2**	2 − 2 = **0**

NOTE:
Priority comparison happens at every time unit
Lower priority number = higher priority