Mohammed Nizamuddin

2503B05144

Lab Task: Round Robin CPU Scheduling (Time Quantum = 3 ms)

A system receives the following processes NOT in arrival order:

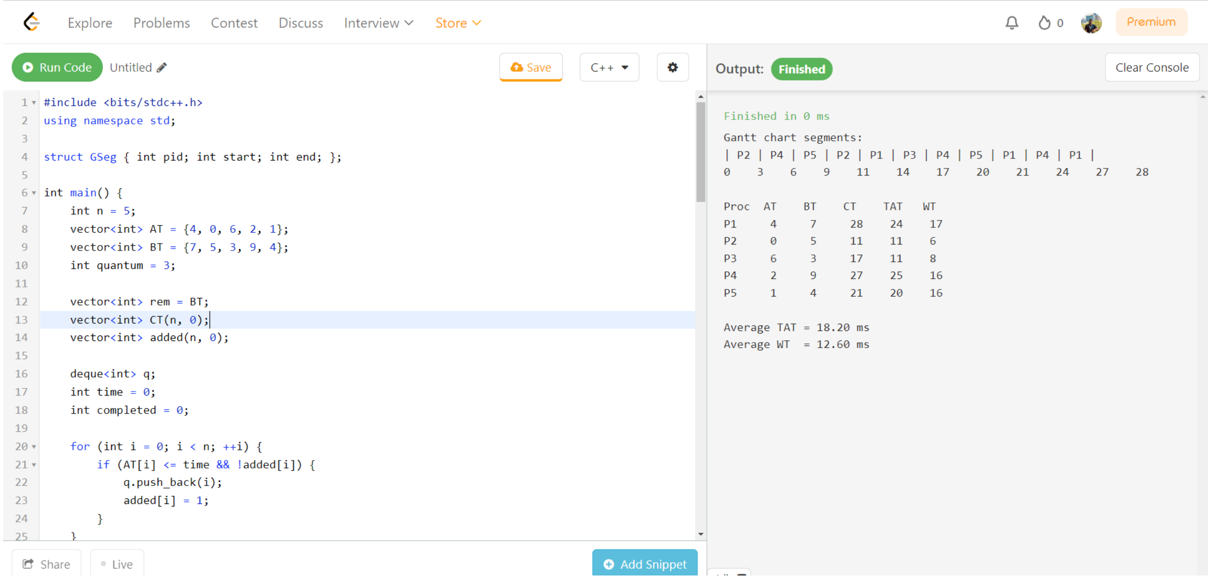
Process	Arrival Time (AT)	Burst Time (BT)
P1	4	7
P2	0	5
P3	6	3
P4	2	9
P5	1	4

Student Tasks

1. Do NOT reorder by arrival time.
   Simulate exactly as given.
2. Apply Round Robin Scheduling with Time Quantum = 3 ms.
3. Draw the complete Gantt Chart.
4. Compute for each process:
   - Completion Time (CT)
   - Turnaround Time (TAT = CT − AT)
   - Waiting Time (WT = TAT − BT)
5. Compute:
   - Average TAT
   - Average WT
6. Write a C program that performs RR scheduling.
7. Compare your manual result with the program output.

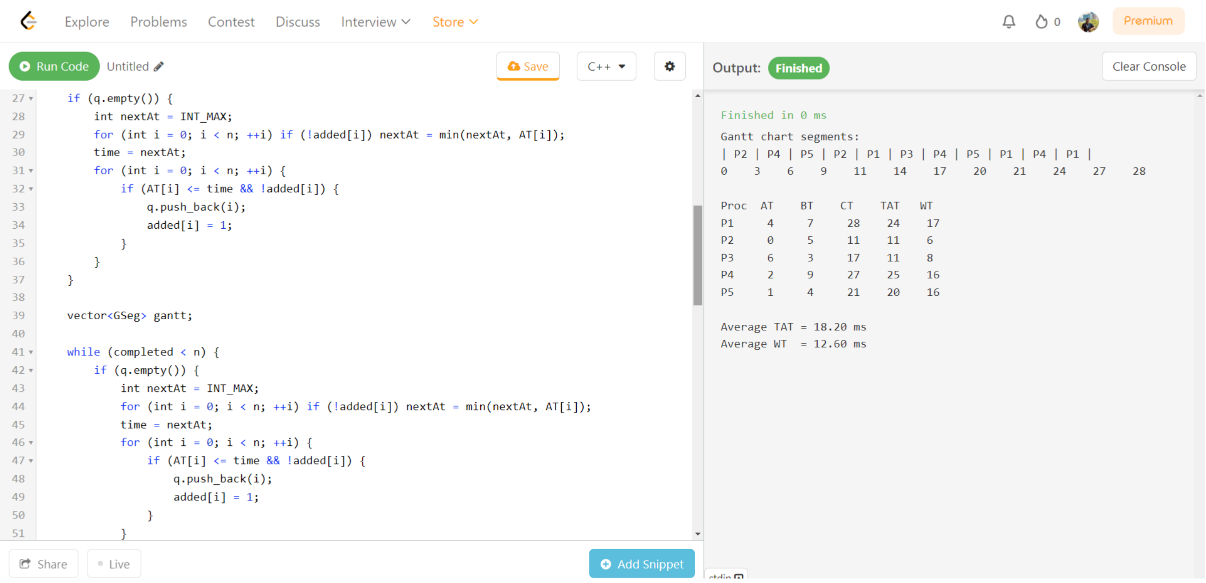
CODE:

```
#include <bits/stdc++.h>
using namespace std;

struct GSeg { int pid; int start; int end; };

int main() {
    int n = 5;
    vector<int> AT = {4, 0, 6, 2, 1};
    vector<int> BT = {7, 5, 3, 9, 4};
    int quantum = 3;

    vector<int> rem = BT;
    vector<int> CT(n, 0);
    vector<int> added(n, 0);

    deque<int> q;
    int time = 0;
    int completed = 0;

    for (int i = 0; i < n; ++i) {
        if (AT[i] <= time && !added[i]) {
            q.push_back(i);
            added[i] = 1;
        }
    }

    if (q.empty()) {
        int nextAt = INT_MAX;
        for (int i = 0; i < n; ++i) if (!added[i]) nextAt = min(nextAt, AT[i]);
        time = nextAt;
        for (int i = 0; i < n; ++i) {
            if (AT[i] <= time && !added[i]) {
                q.push_back(i);
                added[i] = 1;
            }
        }
    }

    vector<GSeg> gantt;

    while (completed < n) {
        if (q.empty()) {
            int nextAt = INT_MAX;
            for (int i = 0; i < n; ++i) if (!added[i]) nextAt = min(nextAt, AT[i]);
            time = nextAt;
            for (int i = 0; i < n; ++i) {
                if (AT[i] <= time && !added[i]) {
                    q.push_back(i);
                    added[i] = 1;
                }
            }
            continue;
        }

        int idx = q.front(); q.pop_front();
        int exec = min(rem[idx], quantum);
        int start = time;
        int end = time + exec;

        rem[idx] -= exec;
        time = end;

        gantt.push_back({idx+1, start, end});

        for (int i = 0; i < n; ++i) {
            if (AT[i] <= time && !added[i]) {
                q.push_back(i);
                added[i] = 1;
            }
        }

        if (rem[idx] == 0) {
            CT[idx] = time;
            completed++;
        } else {
            q.push_back(idx);
        }
    }

    cout << "Gantt chart segments:\n";
    for (auto &s : gantt) cout << "| P" << s.pid << " ";
    cout << "|\n";
    for (auto &s : gantt) cout << s.start << "    ";
    cout << gantt.back().end << "\n\n";

    cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
         << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\n";

    double sumTAT = 0.0, sumWT = 0.0;
    for (int i = 0; i < n; ++i) {
        int tat = CT[i] - AT[i];
        int wt  = tat - BT[i];
        sumTAT += tat;
        sumWT  += wt;
        cout << "P" << (i+1) << setw(5) << ""
             << setw(6) << AT[i] << setw(6) << BT[i]
             << setw(6) << CT[i] << setw(6) << tat << setw(6) << wt << "\n";
    }

    cout << fixed << setprecision(2);
    cout << "\nAverage TAT = " << (sumTAT / n) << " ms\n";
    cout << "Average WT  = " << (sumWT / n)  << " ms\n";

    return 0;
}
```



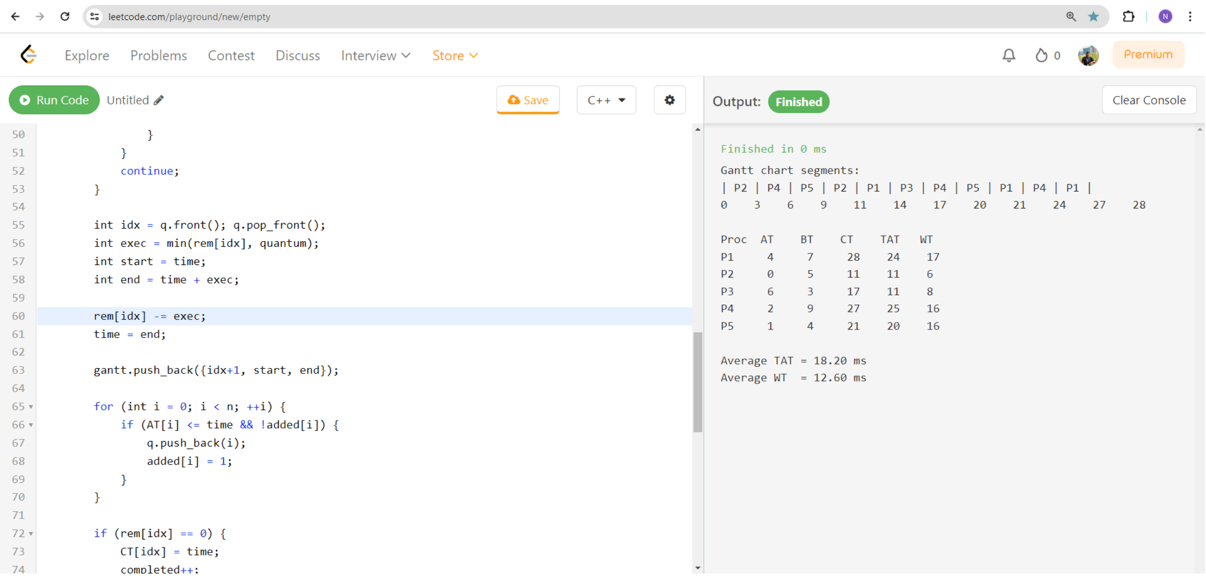
```
#include <bits/stdc++.h>
using namespace std;

struct GSeg { int pid; int start; int end; };

int main() {
    int n = 5;
    vector<int> AT = {4, 0, 6, 2, 1};
    vector<int> BT = {7, 5, 3, 9, 4};
    int quantum = 3;

    vector<int> rem = BT;
    vector<int> CT(n, 0);
    vector<int> added(n, 0);

    deque<int> q;
    int time = 0;
    int completed = 0;

    for (int i = 0; i < n; ++i) {
        if (AT[i] <= time && !added[i]) {
            q.push_back(i);
            added[i] = 1;
        }
    }
```

```
Finished in 0 ms

Gantt chart segments:
| P2 | P4 | P5 | P2 | P1 | P3 | P4 | P5 | P1 | P4 | P1 |
0    3    6    9    11    14    17    20    21    24    27    28

Proc  AT    BT    CT    TAT   WT
P1     4     7     28    24    17
P2     0     5     11    11    6
P3     6     3     17    11    8
P4     2     9     27    25    16
P5     1     4     21    20    16

Average TAT = 18.20 ms
Average WT  = 12.60 ms
```



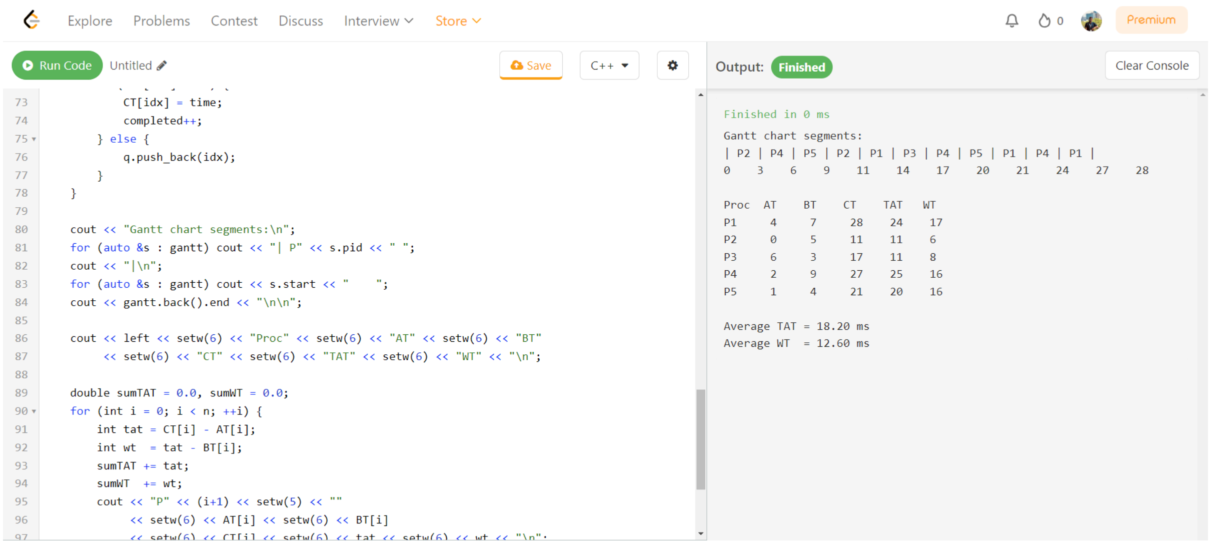
```
    if (q.empty()) {
        int nextAt = INT_MAX;
        for (int i = 0; i < n; ++i) if (!added[i]) nextAt = min(nextAt, AT[i]);
        time = nextAt;
        for (int i = 0; i < n; ++i) {
            if (AT[i] <= time && !added[i]) {
                q.push_back(i);
                added[i] = 1;
            }
        }
    }

    vector<GSeg> gantt;

    while (completed < n) {
        if (q.empty()) {
            int nextAt = INT_MAX;
            for (int i = 0; i < n; ++i) if (!added[i]) nextAt = min(nextAt, AT[i]);
            time = nextAt;
            for (int i = 0; i < n; ++i) {
                if (AT[i] <= time && !added[i]) {
                    q.push_back(i);
                    added[i] = 1;
                }
            }
```

```
Finished in 0 ms

Gantt chart segments:
| P2 | P4 | P5 | P2 | P1 | P3 | P4 | P5 | P1 | P4 | P1 |
0    3    6    9    11    14    17    20    21    24    27    28

Proc  AT    BT    CT    TAT   WT
P1     4     7     28    24    17
P2     0     5     11    11    6
P3     6     3     17    11    8
P4     2     9     27    25    16
P5     1     4     21    20    16

Average TAT = 18.20 ms
Average WT  = 12.60 ms
```



```cpp
                }
            }
            continue;
        }

        int idx = q.front(); q.pop_front();
        int exec = min(rem[idx], quantum);
        int start = time;
        int end = time + exec;

        rem[idx] -= exec;
        time = end;

        gantt.push_back({idx+1, start, end});

        for (int i = 0; i < n; ++i) {
            if (AT[i] <= time && !added[i]) {
                q.push_back(i);
                added[i] = 1;
            }
        }

        if (rem[idx] == 0) {
            CT[idx] = time;
            completed++;
```

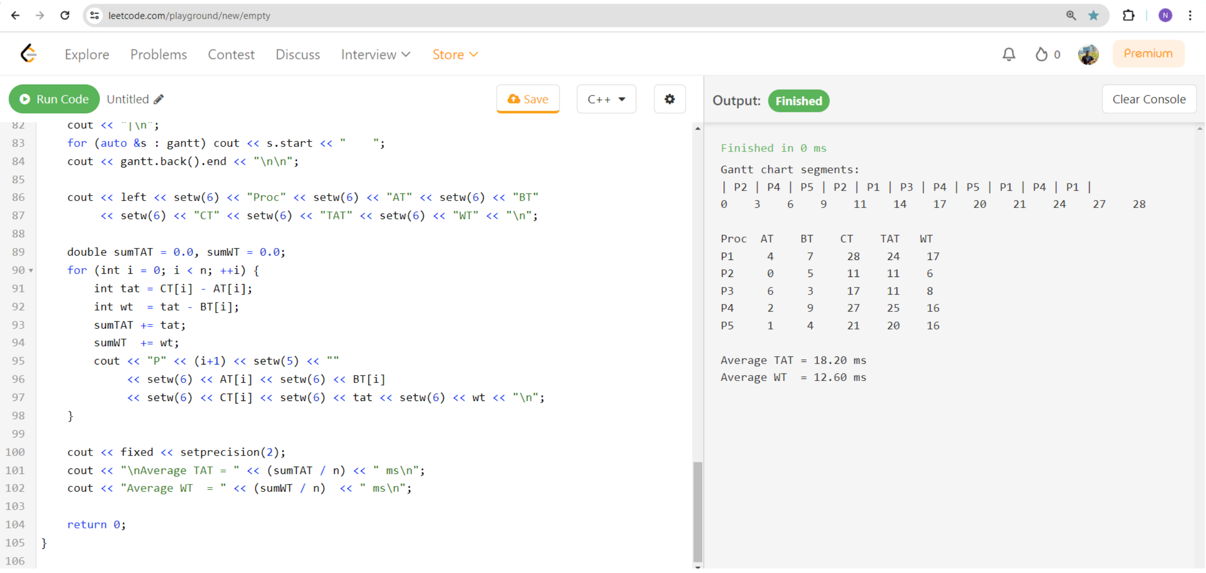
```
Finished in 0 ms

Gantt chart segments:
| P2 | P4 | P5 | P2 | P1 | P3 | P4 | P5 | P1 | P4 | P1 |
0    3    6    9    11    14    17    20    21    24    27    28

Proc  AT    BT    CT    TAT   WT
P1     4     7     28    24    17
P2     0     5     11    11    6
P3     6     3     17    11    8
P4     2     9     27    25    16
P5     1     4     21    20    16

Average TAT = 18.20 ms
Average WT  = 12.60 ms
```



```cpp
            CT[idx] = time;
            completed++;
```

```cpp
        } else {
            q.push_back(idx);
        }
    }

    cout << "Gantt chart segments:\n";
    for (auto &s : gantt) cout << "| P" << s.pid << " ";
    cout << "|\n";
    for (auto &s : gantt) cout << s.start << "    ";
    cout << gantt.back().end << "\n\n";

    cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
         << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\n";

    double sumTAT = 0.0, sumWT = 0.0;
    for (int i = 0; i < n; ++i) {
        int tat = CT[i] - AT[i];
        int wt  = tat - BT[i];
        sumTAT += tat;
        sumWT  += wt;
        cout << "P" << (i+1) << setw(5) << ""
             << setw(6) << AT[i] << setw(6) << BT[i]
             << setw(6) << CT[i] << setw(6) << tat << setw(6) << wt << "\n";
```

```
Finished in 0 ms

Gantt chart segments:
| P2 | P4 | P5 | P2 | P1 | P3 | P4 | P5 | P1 | P4 | P1 |
0    3    6    9    11    14    17    20    21    24    27    28

Proc  AT    BT    CT    TAT   WT
P1     4     7     28    24    17
P2     0     5     11    11    6
P3     6     3     17    11    8
P4     2     9     27    25    16
P5     1     4     21    20    16

Average TAT = 18.20 ms
Average WT  = 12.60 ms
```



```cpp
    cout << "|\n";
    for (auto &s : gantt) cout << s.start << "    ";
    cout << gantt.back().end << "\n\n";

    cout << left << setw(6) << "Proc" << setw(6) << "AT" << setw(6) << "BT"
         << setw(6) << "CT" << setw(6) << "TAT" << setw(6) << "WT" << "\n";

    double sumTAT = 0.0, sumWT = 0.0;
    for (int i = 0; i < n; ++i) {
        int tat = CT[i] - AT[i];
        int wt  = tat - BT[i];
        sumTAT += tat;
        sumWT  += wt;
        cout << "P" << (i+1) << setw(5) << ""
             << setw(6) << AT[i] << setw(6) << BT[i]
             << setw(6) << CT[i] << setw(6) << tat << setw(6) << wt << "\n";
```

```cpp
    }

    cout << fixed << setprecision(2);
    cout << "\nAverage TAT = " << (sumTAT / n) << " ms\n";
    cout << "Average WT  = " << (sumWT / n)  << " ms\n";

    return 0;
}

```

```
Finished in 0 ms

Gantt chart segments:
| P2 | P4 | P5 | P2 | P1 | P3 | P4 | P5 | P1 | P4 | P1 |
0    3    6    9    11    14    17    20    21    24    27    28

Proc  AT    BT    CT    TAT   WT
P1     4     7     28    24    17
P2     0     5     11    11    6
P3     6     3     17    11    8
P4     2     9     27    25    16
P5     1     4     21    20    16

Average TAT = 18.20 ms
Average WT  = 12.60 ms
```