

**ROUND ROBIN****OUTPUT:**

```

Enter number of processes: 3
Enter Process 1 ID (char): A
Enter Arrival Time: 0
Enter Burst Time: 4
Enter Process 2 ID (char): B
Enter Arrival Time: 2
Enter Burst Time: 2
Enter Process 3 ID (char): C
Enter Arrival Time: 3
Enter Burst Time: 2
Enter Time Quantum: 2

ID      AT      BT      CT      TT      WT
A       0       4       6       6       2
B       2       2       4       2       0
C       3       2       8       5       3

Avg WT = 1.66667
Avg TT = 4.33333

Gantt Chart:
| A | B | A | C |
0   2   4   6   8

```

**EXPLANATION/COMPUTATION**

TIME	EXPLANATION
0	ARRIVES A
2	A GO TO READY QUEUE BECAUSE B WILL RUN. RQ[A]
4	B FINISHED RUN A. QR: [NULL]
6	A FINISHED RUN C
C	FINISHED



```

1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int n, i, j, time = 0, done = 0, quantum;
6     char id[20];
7     int at[20];
8     int bt[20];
9     int rt[20];
10    int ct[20], tt[20], wt[20];
11    int gantt[200], gTime[201], gLen = 0;
12    int queue[100];
13    int front = 0, rear = 0;
14    bool inQueue[20] = {false};
15    float totWT = 0, totTT = 0;
16
17    cout << "Enter number of processes: ";
18    cin >> n;
19    for (i = 0; i < n; i++)
20    {
21        cout << "Enter Process " << i + 1 << " ID (char): ";
22        cin >> id[i];
23        cout << "Enter Arrival Time: ";
24        cin >> at[i];
25        cout << "Enter Burst Time: ";
26        cin >> bt[i];
27        rt[i] = bt[i];
28    }
29    cout << "Enter Time Quantum: ";
30    cin >> quantum;
31
32    queue[rear++] = 0;
33    inQueue[0] = true;
34    time = at[0];
35    while (done < n)
36    {
37        if (front == rear)
38        {
39            for (i = 0; i < n; i++)
40            {
41                if (!inQueue[i] && rt[i] > 0)
42                {
43                    queue[rear++] = i;
44                    inQueue[i] = true;
45                    time = at[i];
46                    break;
47                }
48            }
49            if (front == rear)
50            {
51                time++;
52                continue;
53            }
54        }
55
56        int idx = queue[front++];
57
58        int execTime = 0;
59        if (rt[idx] > quantum)
60        {
61            execTime = quantum;
62        }
63        else
64        {
65            execTime = rt[idx];
66        }
67
68        if (gLen == 0 || gantt[glen - 1] != idx)
69        {
70            eTime[glen] = time;
71            gTime[glen] = time;
72            gantt[glen] = idx;
73
74            rt[idx] -= execTime;
75            time += execTime;
76
77            for (i = 0; i < n; i++)
78            {
79                if (!inQueue[i] && at[i] <= time && rt[i] > 0)
80                {
81                    queue[rear++] = i;
82                    inQueue[i] = true;
83                }
84            }
85            if (rt[idx] == 0)
86            {
87                ct[idx] = time;
88                tt[idx] = ct[idx] - at[idx];
89                wt[idx] = tt[idx] - bt[idx];
90                totWT += wt[idx];
91                totTT += tt[idx];
92                done++;
93            }
94            else
95            {
96                queue[rear++] = idx;
97            }
98        }
99        gTime[gLen] = time;
100
101    cout << "\nID\tAT\tBT\tCT\tTT\tWT\n";
102    for (i = 0; i < n; i++)
103    {
104        cout << id[i] << "\t" << at[i] << "\t" << bt[i] << "\t"
105        << ct[i] << "\t" << tt[i] << "\t" << wt[i] << "\n";
106    }
107    cout << "\nAvg WT = " << totWT / n;
108    cout << "\nAvg TT = " << totTT / n;
109
110    cout << "\n\nGantt Chart:\n" ;
111    for (i = 0; i < gLen; i++)
112    {
113        cout << id[gantt[i]] << " | ";
114    }
115    cout << "\n";
116    for (i = 0; i <= gLen; i++)
117    {
118        cout << gTime[i] << "   ";
119    }
120    cout << "\n";
121
122    return 0;
123 }
```

```

55
56     int idx = queue[front++];
57
58     int execTime = 0;
59     if (rt[idx] > quantum)
60     {
61         execTime = quantum;
62     }
63     else
64     {
65         execTime = rt[idx];
66     }
67
68     if (gLen == 0 || gantt[glen - 1] != idx)
69     {
70         gTime[glen] = time;
71         gantt[glen] = idx;
72     }
73
74     rt[idx] -= execTime;
75     time += execTime;
76
77     for (i = 0; i < n; i++)
78     {
79         if (!inQueue[i] && at[i] <= time && rt[i] > 0)
80         {
81             queue[rear++] = i;
82             inQueue[i] = true;
83         }
84     }
85     if (rt[idx] == 0)
86     {
87         ct[idx] = time;
88         tt[idx] = ct[idx] - at[idx];
89         wt[idx] = tt[idx] - bt[idx];
90         totWT += wt[idx];
91         totTT += tt[idx];
92         done++;
93     }
94     else
95     {
96         queue[rear++] = idx;
97     }
98 }
99 gTime[gLen] = time;
100
101 cout << "\nID\tAT\tBT\tCT\tTT\tWT\n";
102 for (i = 0; i < n; i++)
103 {
104     cout << id[i] << "\t" << at[i] << "\t" << bt[i] << "\t"
105     << ct[i] << "\t" << tt[i] << "\t" << wt[i] << "\n";
106 }
107 cout << "\nAvg WT = " << totWT / n;
108 cout << "\nAvg TT = " << totTT / n;
109
110 cout << "\n\nGantt Chart:\n" ;
111 for (i = 0; i < gLen; i++)
112 {
113     cout << id[gantt[i]] << " | ";
114 }
115 cout << "\n";
116 for (i = 0; i <= gLen; i++)
117 {
118     cout << gTime[i] << "   ";
119 }
120 cout << "\n";
121
122 }
```

