Program: Round Robin Scheduling Without Arrival Time.

CODE:

```
#include #include using namespace std;

struct Process { int id, bt, remaining_bt, wt, tat; };

void roundRobin(Process p[], int n, int quantum) { queue q; for (int i
= 0; i < n; i++) q.push(i); // Enqueue all processes initially

int time = 0;
float total_wt = 0, total_tat = 0;

while (!q.empty()) {
    int i = q.front();
    q.pop();

    if (p[i].remaining_bt > quantum) {
        time += quantum;
        p[i].remaining_bt -= quantum;
        q.push(i); // Re-add process to queue if not finished
    } else {
        time += p[i].remaining_bt;
        p[i].tat = time;  // Turnaround Time = Completion Time (since
AT = 0)
        p[i].wt = p[i].tat - p[i].bt; // Waiting Time = TAT - BT
        total_wt += p[i].wt;
        total_tat += p[i].tat;
    }
}
cout << "\nProcess\tBT\tWT\tTAT\n";
for (int i = 0; i < n; i++) {
    cout << p[i].id << "\t" << p[i].bt << "\t" << p[i].wt << "\t" <<
p[i].tat << endl;
}

cout << "\nAverage Waiting Time: " << (total_wt / n);
cout << "\nAverage Turnaround Time: " << (total_tat / n) << endl;
 }
```

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

Process p[n];

for (int i = 0; i < n; i++) {
    cout << "Enter burst time for process " << i + 1 << ": ";
    cin >> p[i].bt;
    p[i].id = i + 1;
    p[i].remaining_bt = p[i].bt; // Initialize remaining burst time
}

cout << "Enter time quantum: ";
cin >> quantum;

roundRobin(p, n, quantum);
return 0;
 }
```

OUTPUT:

Enter number of processes: 3

Enter burst time for process 1: 5

Enter burst time for process 2: 3

Enter burst time for process 3: 8

Enter time quantum: 2

Process	BT	WT	TAT
1	5	7	12
2	3	6	9
3	8	8	16

Average Waiting Time: 7

Average Turnaround Time: 12.3333