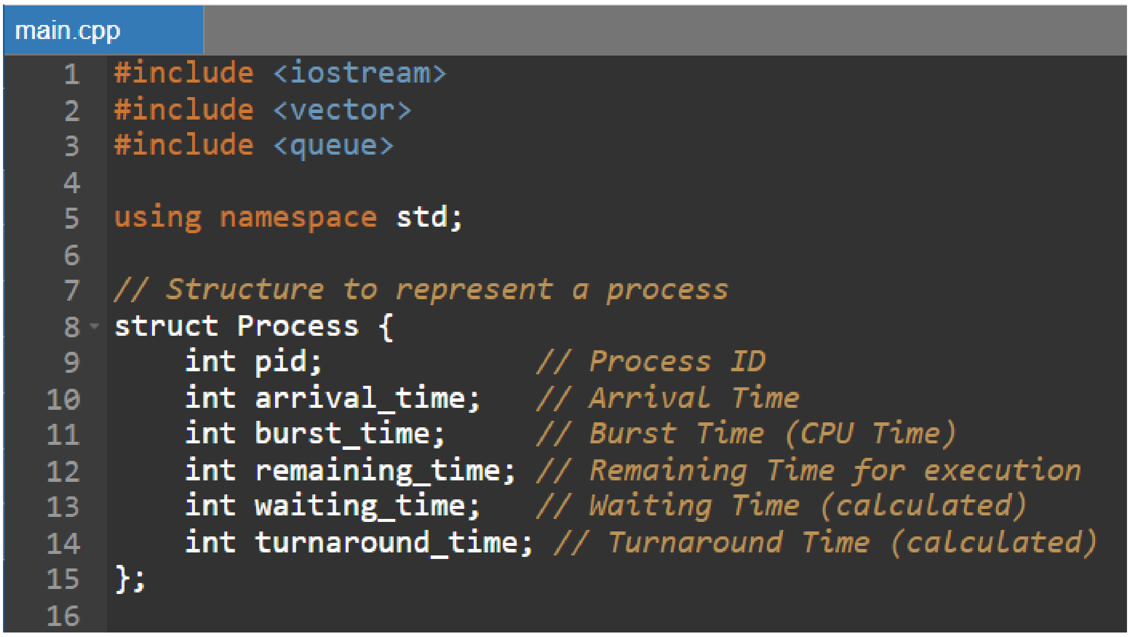
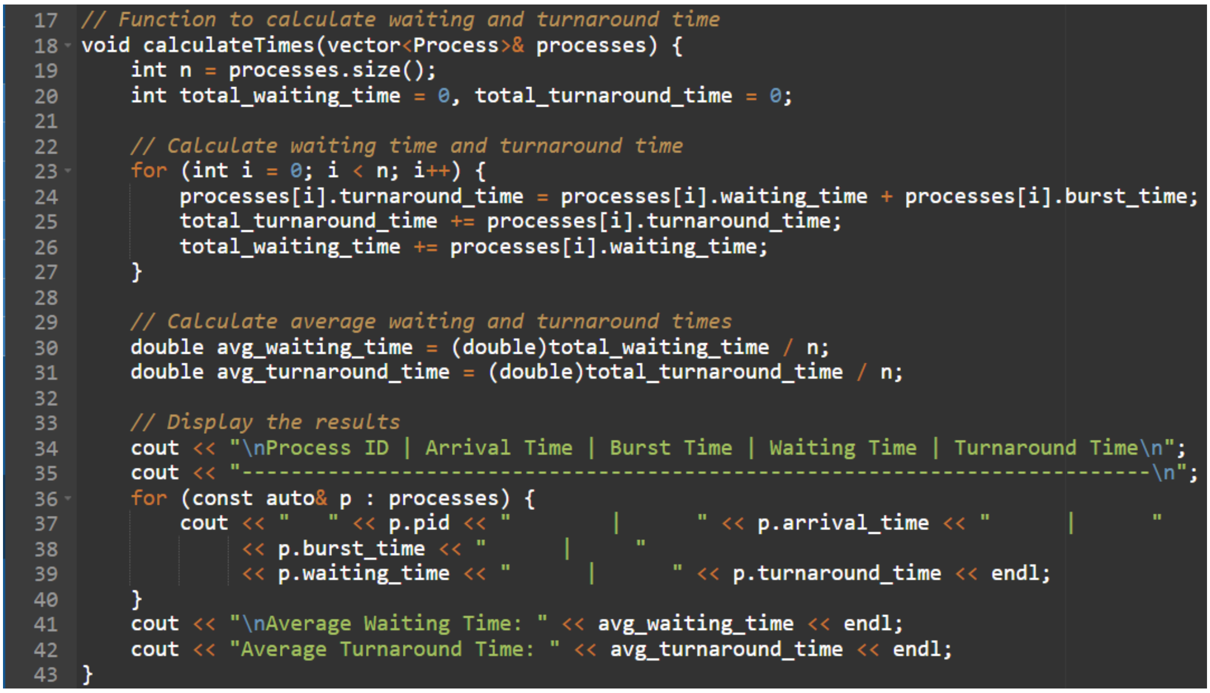
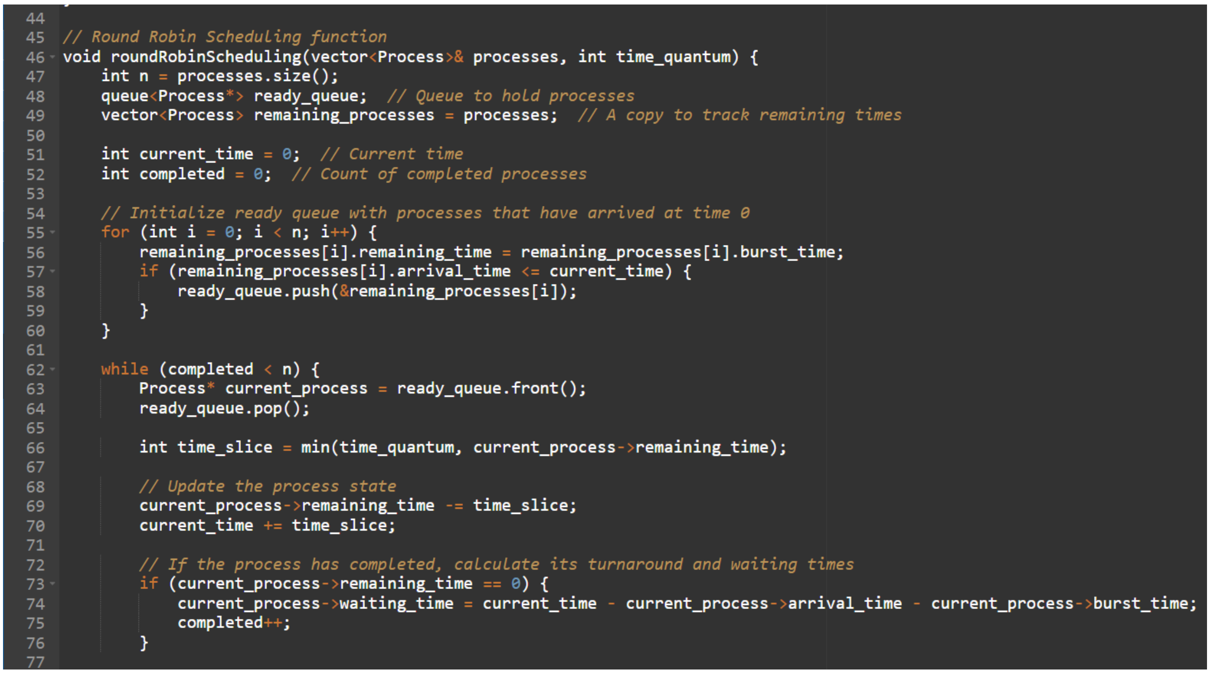


main.cpp

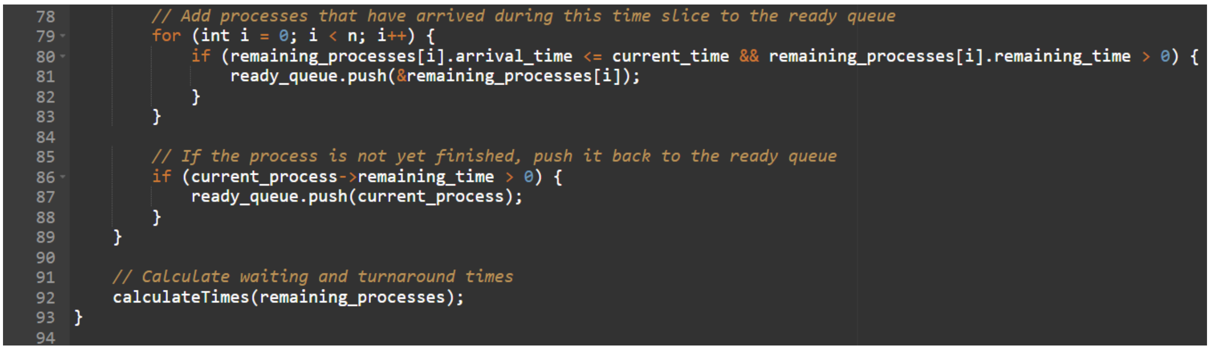
```cpp
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

// Structure to represent a process
struct Process {
    int pid;            // Process ID
    int arrival_time;   // Arrival Time
    int burst_time;     // Burst Time (CPU Time)
    int remaining_time; // Remaining Time for execution
    int waiting_time;   // Waiting Time (calculated)
    int turnaround_time; // Turnaround Time (calculated)
};

```



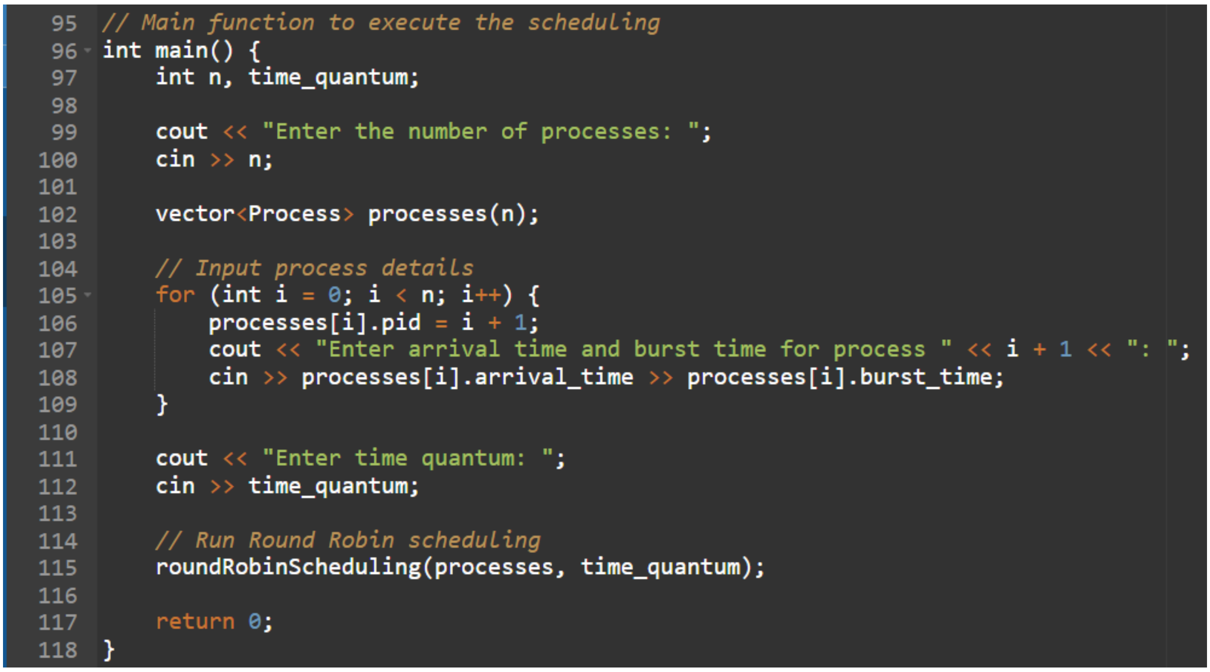
```cpp
// Function to calculate waiting and turnaround time
void calculateTimes(vector<Process>& processes) {
    int n = processes.size();
    int total_waiting_time = 0, total_turnaround_time = 0;

    // Calculate waiting time and turnaround time
    for (int i = 0; i < n; i++) {
        processes[i].turnaround_time = processes[i].waiting_time + processes[i].burst_time;
        total_turnaround_time += processes[i].turnaround_time;
        total_waiting_time += processes[i].waiting_time;
    }

    // Calculate average waiting and turnaround times
    double avg_waiting_time = (double)total_waiting_time / n;
    double avg_turnaround_time = (double)total_turnaround_time / n;

    // Display the results
    cout << "\nProcess ID | Arrival Time | Burst Time | Waiting Time | Turnaround Time\n";
    cout << "---------------------------------------------------------------------------\n";
    for (const auto& p : processes) {
        cout << "   " << p.pid << "        |      " << p.arrival_time << "      |      "
             << p.burst_time << "      |     "
             << p.waiting_time << "      |      " << p.turnaround_time << endl;
    }
    cout << "\nAverage Waiting Time: " << avg_waiting_time << endl;
    cout << "Average Turnaround Time: " << avg_turnaround_time << endl;
}
```



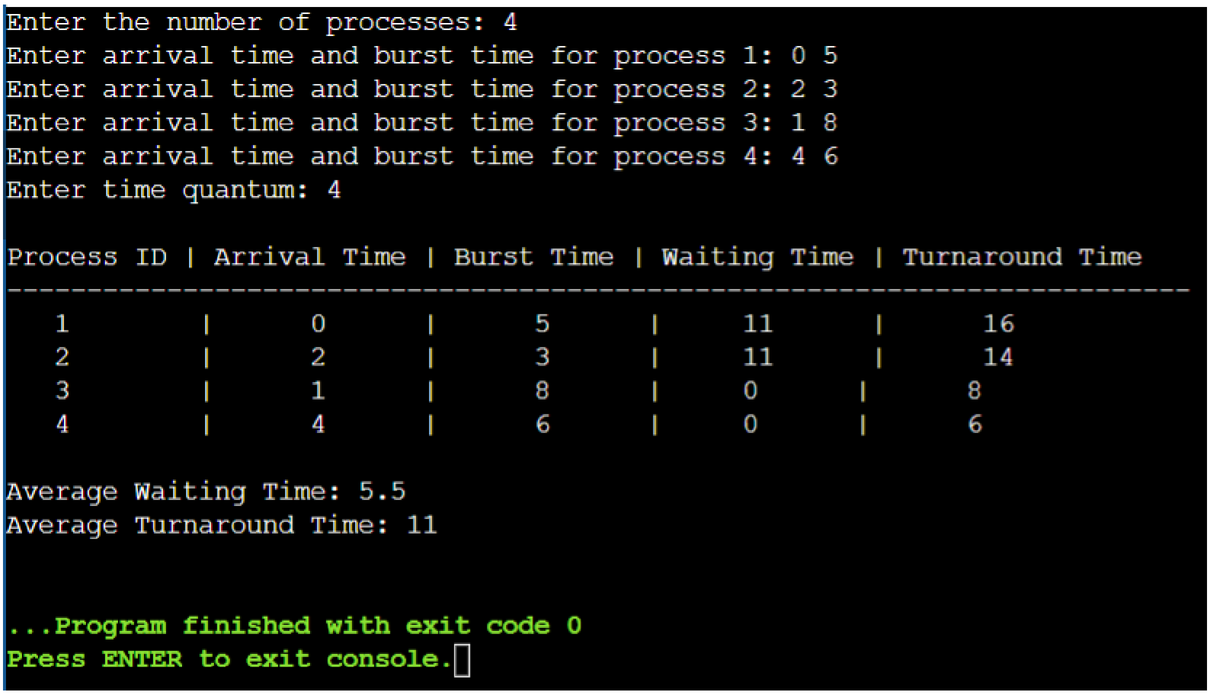
```cpp

// Round Robin Scheduling function
void roundRobinScheduling(vector<Process>& processes, int time_quantum) {
    int n = processes.size();
    queue<Process*> ready_queue;  // Queue to hold processes
    vector<Process> remaining_processes = processes;  // A copy to track remaining times

    int current_time = 0;  // Current time
    int completed = 0;  // Count of completed processes

    // Initialize ready queue with processes that have arrived at time 0
    for (int i = 0; i < n; i++) {
        remaining_processes[i].remaining_time = remaining_processes[i].burst_time;
        if (remaining_processes[i].arrival_time <= current_time) {
            ready_queue.push(&remaining_processes[i]);
        }
    }

    while (completed < n) {
        Process* current_process = ready_queue.front();
        ready_queue.pop();

        int time_slice = min(time_quantum, current_process->remaining_time);

        // Update the process state
        current_process->remaining_time -= time_slice;
        current_time += time_slice;

        // If the process has completed, calculate its turnaround and waiting times
        if (current_process->remaining_time == 0) {
            current_process->waiting_time = current_time - current_process->arrival_time - current_process->burst_time;
            completed++;
        }

```



```cpp
        // Add processes that have arrived during this time slice to the ready queue
        for (int i = 0; i < n; i++) {
            if (remaining_processes[i].arrival_time <= current_time && remaining_processes[i].remaining_time > 0) {
                ready_queue.push(&remaining_processes[i]);
            }
        }

        // If the process is not yet finished, push it back to the ready queue
        if (current_process->remaining_time > 0) {
            ready_queue.push(current_process);
        }
    }

    // Calculate waiting and turnaround times
    calculateTimes(remaining_processes);
}

```



```cpp
// Main function to execute the scheduling
int main() {
    int n, time_quantum;

    cout << "Enter the number of processes: ";
    cin >> n;

    vector<Process> processes(n);

    // Input process details
    for (int i = 0; i < n; i++) {
        processes[i].pid = i + 1;
        cout << "Enter arrival time and burst time for process " << i + 1 << ": ";
        cin >> processes[i].arrival_time >> processes[i].burst_time;
    }

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

    // Run Round Robin scheduling
    roundRobinScheduling(processes, time_quantum);

    return 0;
}
```



```
Enter the number of processes: 4
Enter arrival time and burst time for process 1: 0 5
Enter arrival time and burst time for process 2: 2 3
Enter arrival time and burst time for process 3: 1 8
Enter arrival time and burst time for process 4: 4 6
Enter time quantum: 4

Process ID | Arrival Time | Burst Time | Waiting Time | Turnaround Time
--------------------------------------------------------------------------
   1        |      0      |      5      |     11      |      16
   2        |      2      |      3      |     11      |      14
   3        |      1      |      8      |     0      |      8
   4        |      4      |      6      |     0      |      6

Average Waiting Time: 5.5
Average Turnaround Time: 11


...Program finished with exit code 0
Press ENTER to exit console.
```