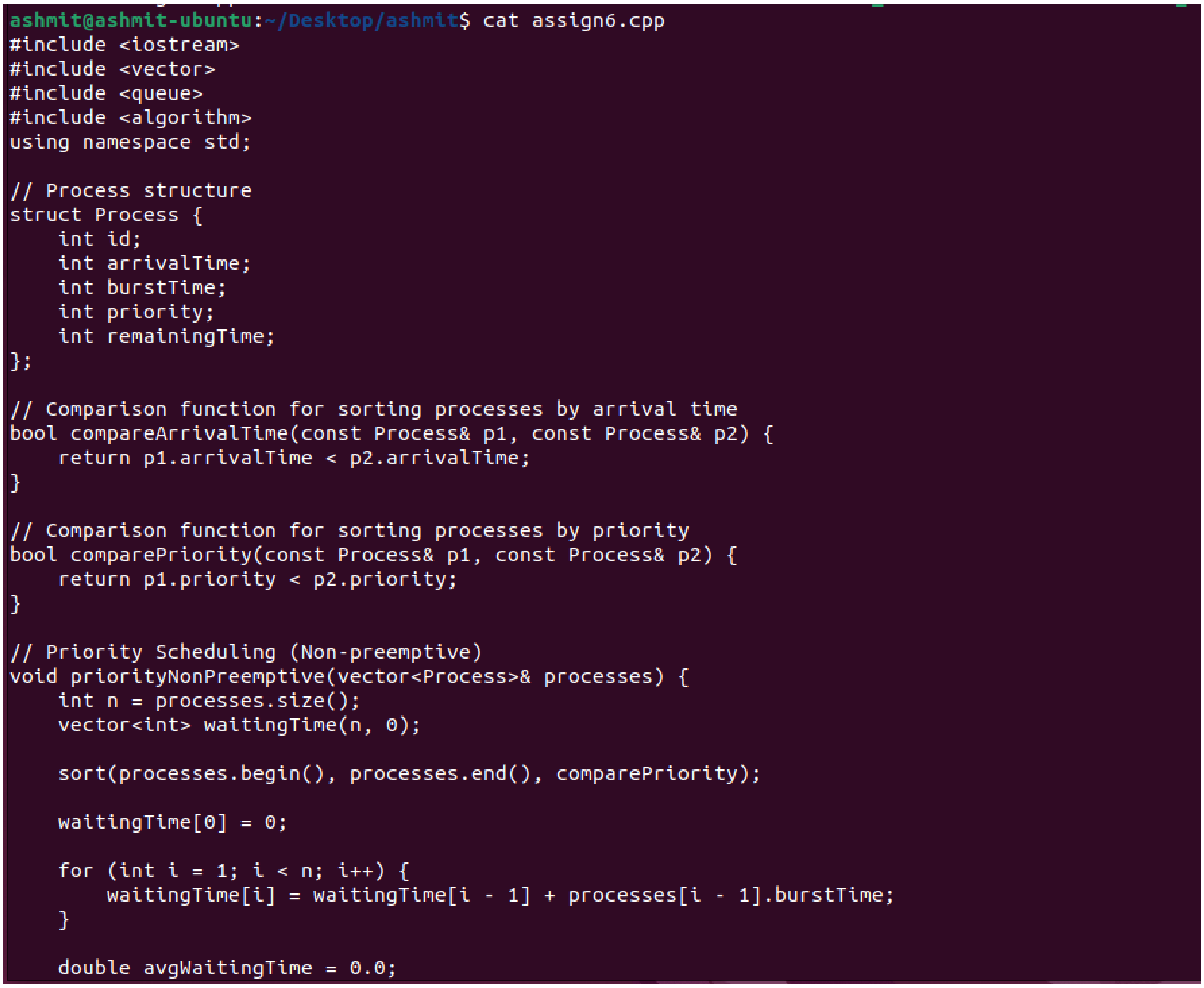
Name: Ashmit Thawait

Roll No: 102203790

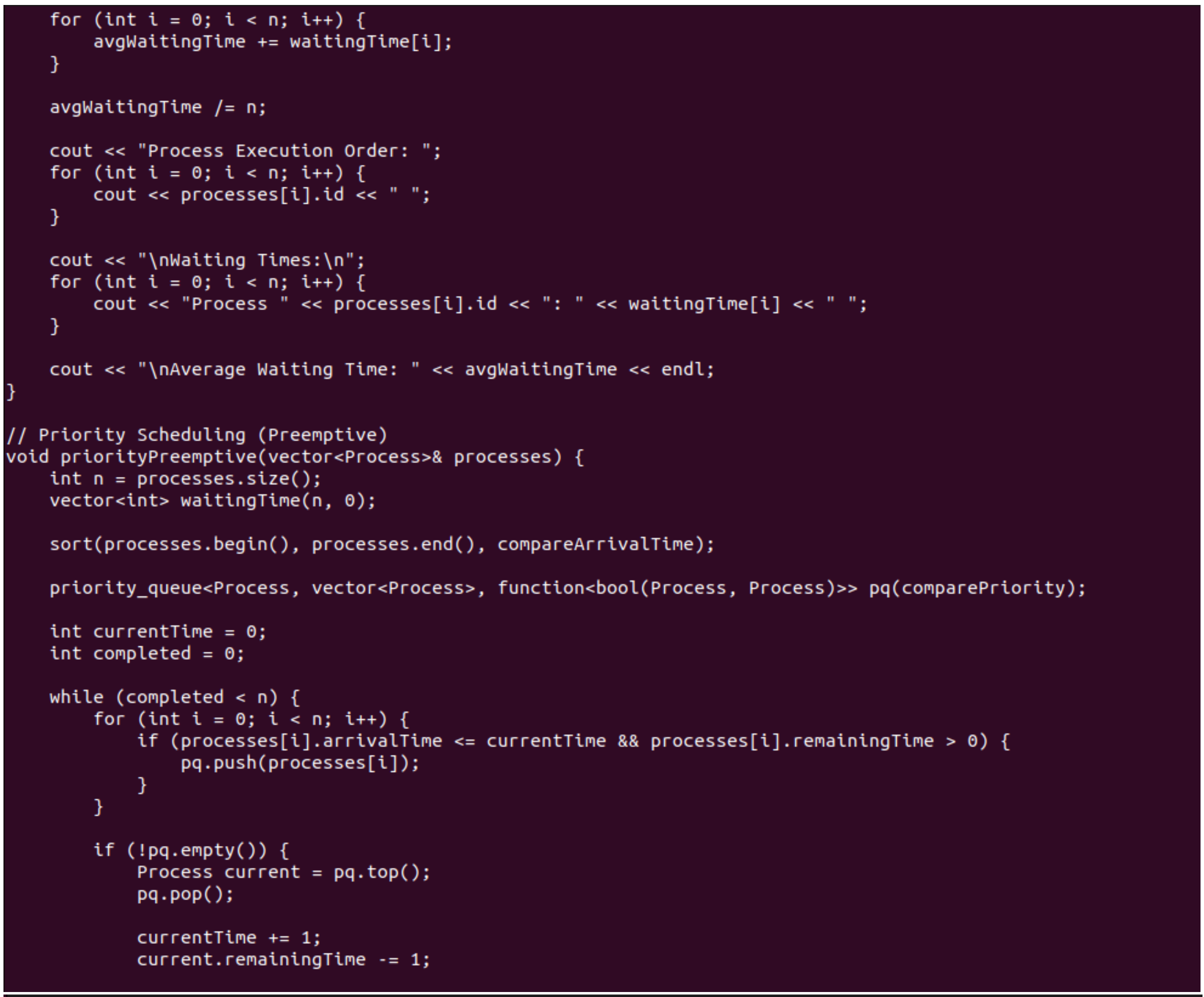
Sub-group: 2CO-17

OS Assignment-6

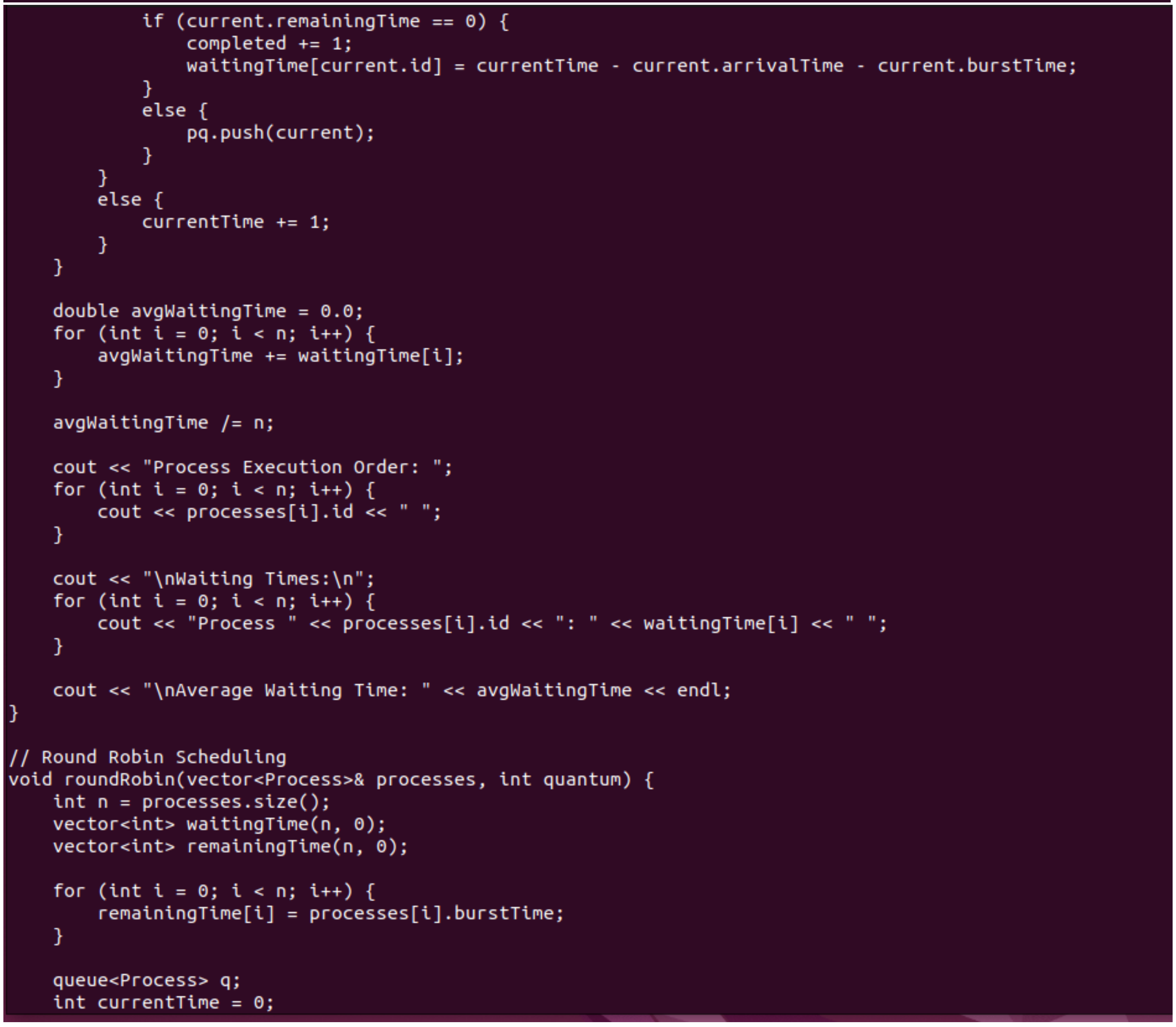
Write a program using C/C++/Java to simulate the priority scheduling (pre-emptive as well as non-preemptive approach) and RR, CPU scheduling algorithms. The scenario is: user may input n processes with respective CPU burst time and arrival time (also take the priority number in case of priority scheduling). System will ask the user to select the type of algorithm from the list mentioned above. System should display the waiting time for each process, average waiting time for the whole system, and final execution sequence.



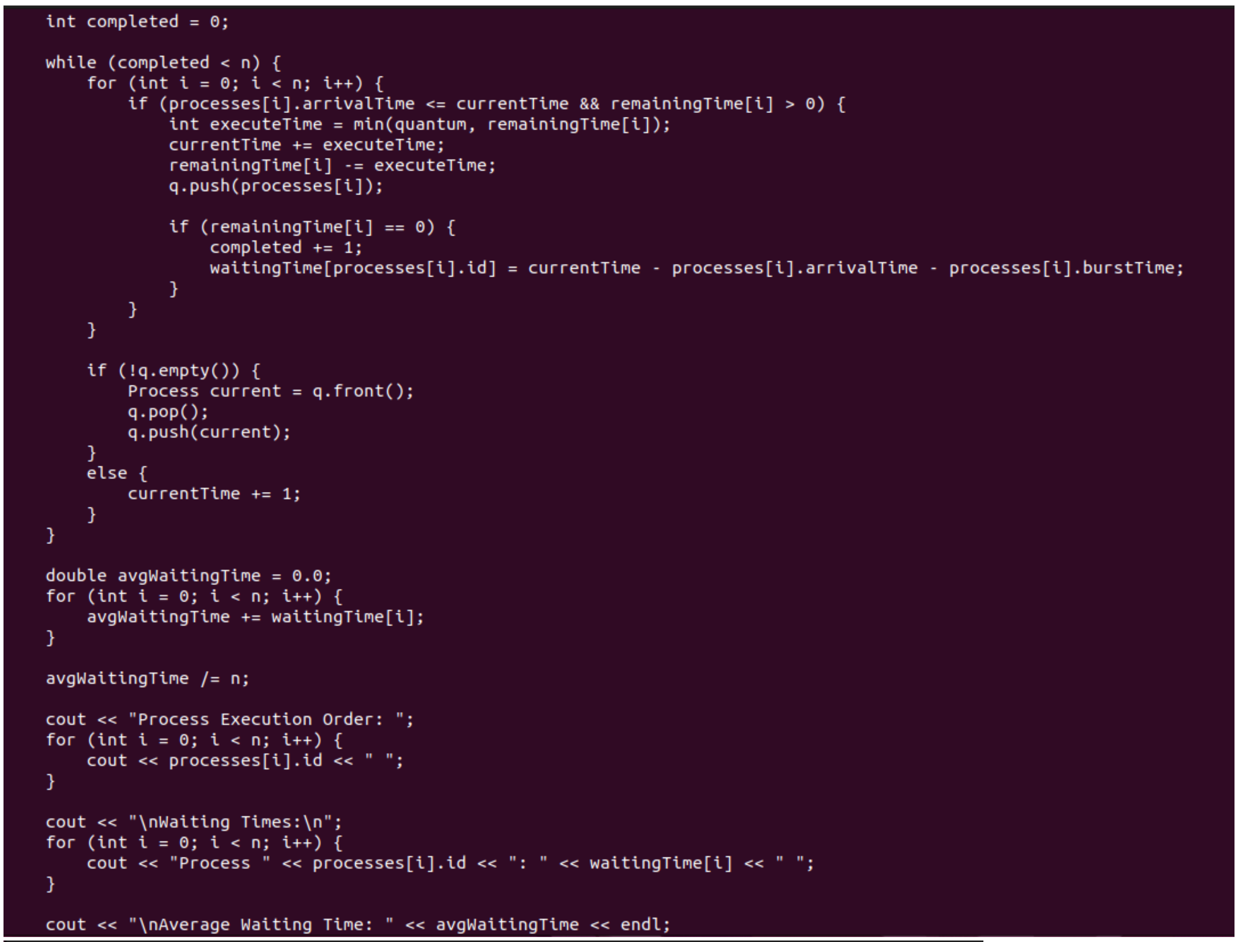
```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ cat assign6.cpp
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

// Process structure
struct Process {
    int id;
    int arrivalTime;
    int burstTime;
    int priority;
    int remainingTime;
};

// Comparison function for sorting processes by arrival time
bool compareArrivalTime(const Process& p1, const Process& p2) {
    return p1.arrivalTime < p2.arrivalTime;
}

// Comparison function for sorting processes by priority
bool comparePriority(const Process& p1, const Process& p2) {
    return p1.priority < p2.priority;
}

// Priority Scheduling (Non-preemptive)
void priorityNonPreemptive(vector<Process>& processes) {
    int n = processes.size();
    vector<int> waitingTime(n, 0);

    sort(processes.begin(), processes.end(), comparePriority);

    waitingTime[0] = 0;

    for (int i = 1; i < n; i++) {
        waitingTime[i] = waitingTime[i - 1] + processes[i - 1].burstTime;
    }

    double avgWaitingTime = 0.0;
```



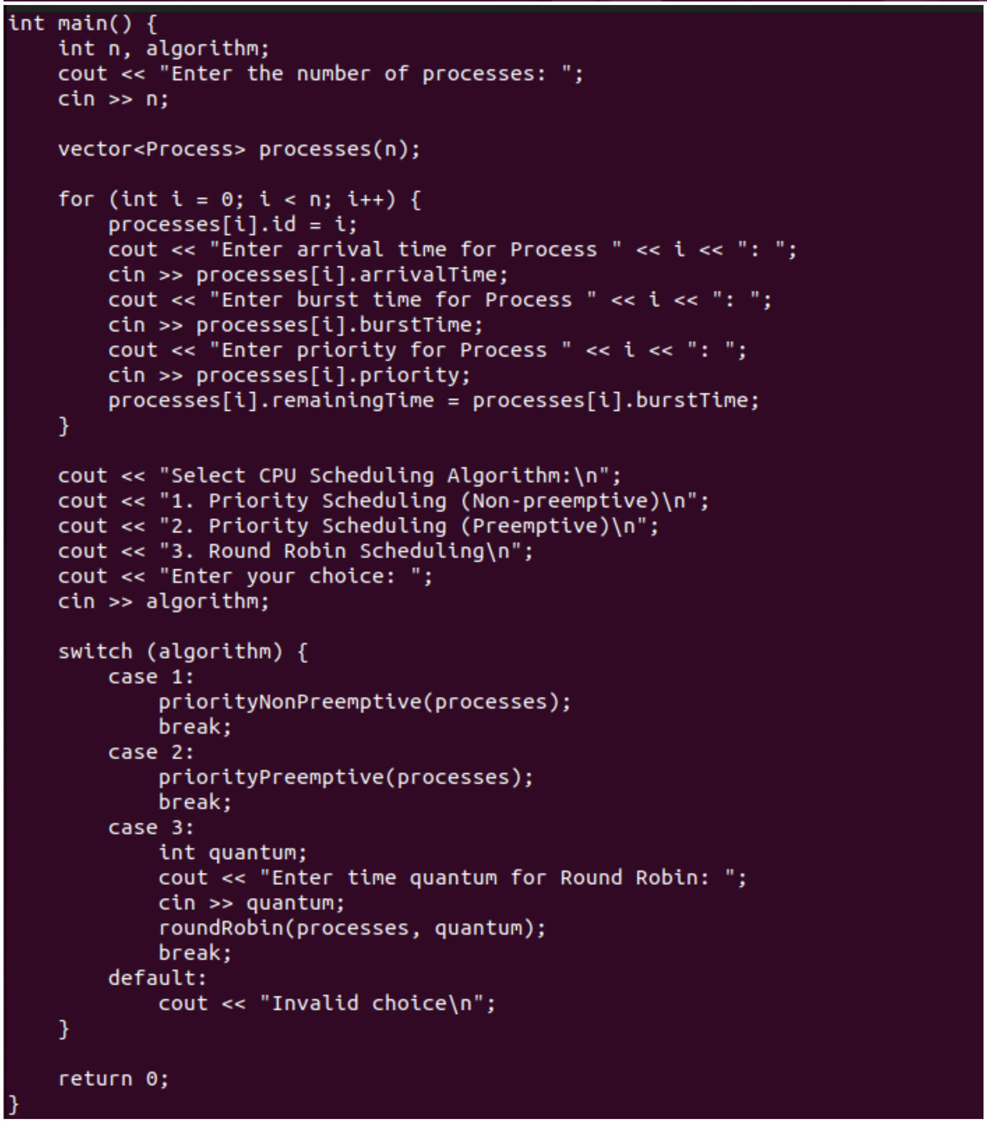
```cpp
    for (int i = 0; i < n; i++) {
        avgWaitingTime += waitingTime[i];
    }

    avgWaitingTime /= n;

    cout << "Process Execution Order: ";
    for (int i = 0; i < n; i++) {
        cout << processes[i].id << " ";
    }

    cout << "\nWaiting Times:\n";
    for (int i = 0; i < n; i++) {
        cout << "Process " << processes[i].id << ": " << waitingTime[i] << " ";
    }

    cout << "\nAverage Waiting Time: " << avgWaitingTime << endl;
}

// Priority Scheduling (Preemptive)
void priorityPreemptive(vector<Process>& processes) {
    int n = processes.size();
    vector<int> waitingTime(n, 0);

    sort(processes.begin(), processes.end(), compareArrivalTime);

    priority_queue<Process, vector<Process>, function<bool(Process, Process)>> pq(comparePriority);

    int currentTime = 0;
    int completed = 0;

    while (completed < n) {
        for (int i = 0; i < n; i++) {
            if (processes[i].arrivalTime <= currentTime && processes[i].remainingTime > 0) {
                pq.push(processes[i]);
            }
        }

        if (!pq.empty()) {
            Process current = pq.top();
            pq.pop();

            currentTime += 1;
            current.remainingTime -= 1;
```



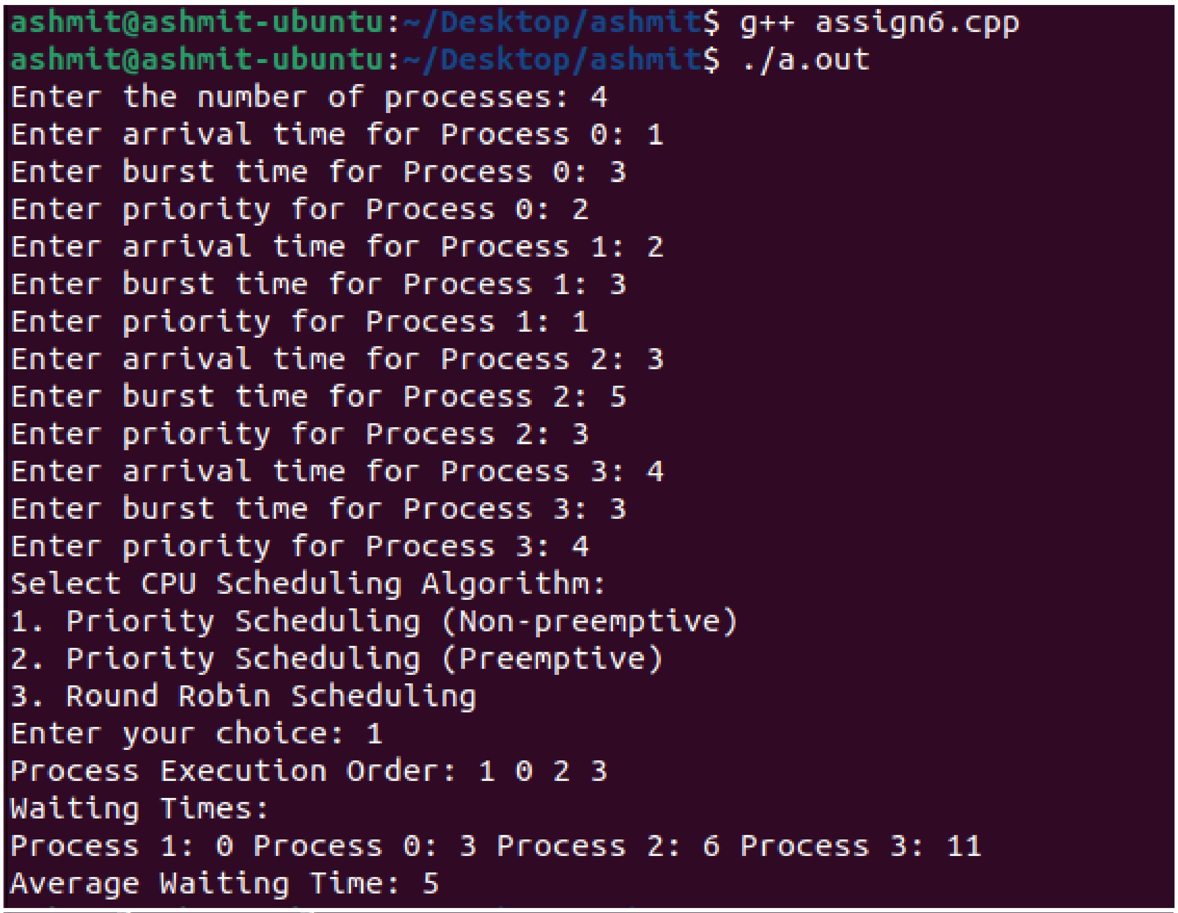
```cpp
            if (current.remainingTime == 0) {
                completed += 1;
                waitingTime[current.id] = currentTime - current.arrivalTime - current.burstTime;
            }
            else {
                pq.push(current);
            }
        }
        else {
            currentTime += 1;
        }
    }

    double avgWaitingTime = 0.0;
    for (int i = 0; i < n; i++) {
        avgWaitingTime += waitingTime[i];
    }

    avgWaitingTime /= n;

    cout << "Process Execution Order: ";
    for (int i = 0; i < n; i++) {
        cout << processes[i].id << " ";
    }

    cout << "\nWaiting Times:\n";
    for (int i = 0; i < n; i++) {
        cout << "Process " << processes[i].id << ": " << waitingTime[i] << " ";
    }

    cout << "\nAverage Waiting Time: " << avgWaitingTime << endl;
}

// Round Robin Scheduling
void roundRobin(vector<Process>& processes, int quantum) {
    int n = processes.size();
    vector<int> waitingTime(n, 0);
    vector<int> remainingTime(n, 0);

    for (int i = 0; i < n; i++) {
        remainingTime[i] = processes[i].burstTime;
    }

    queue<Process> q;
    int currentTime = 0;
```



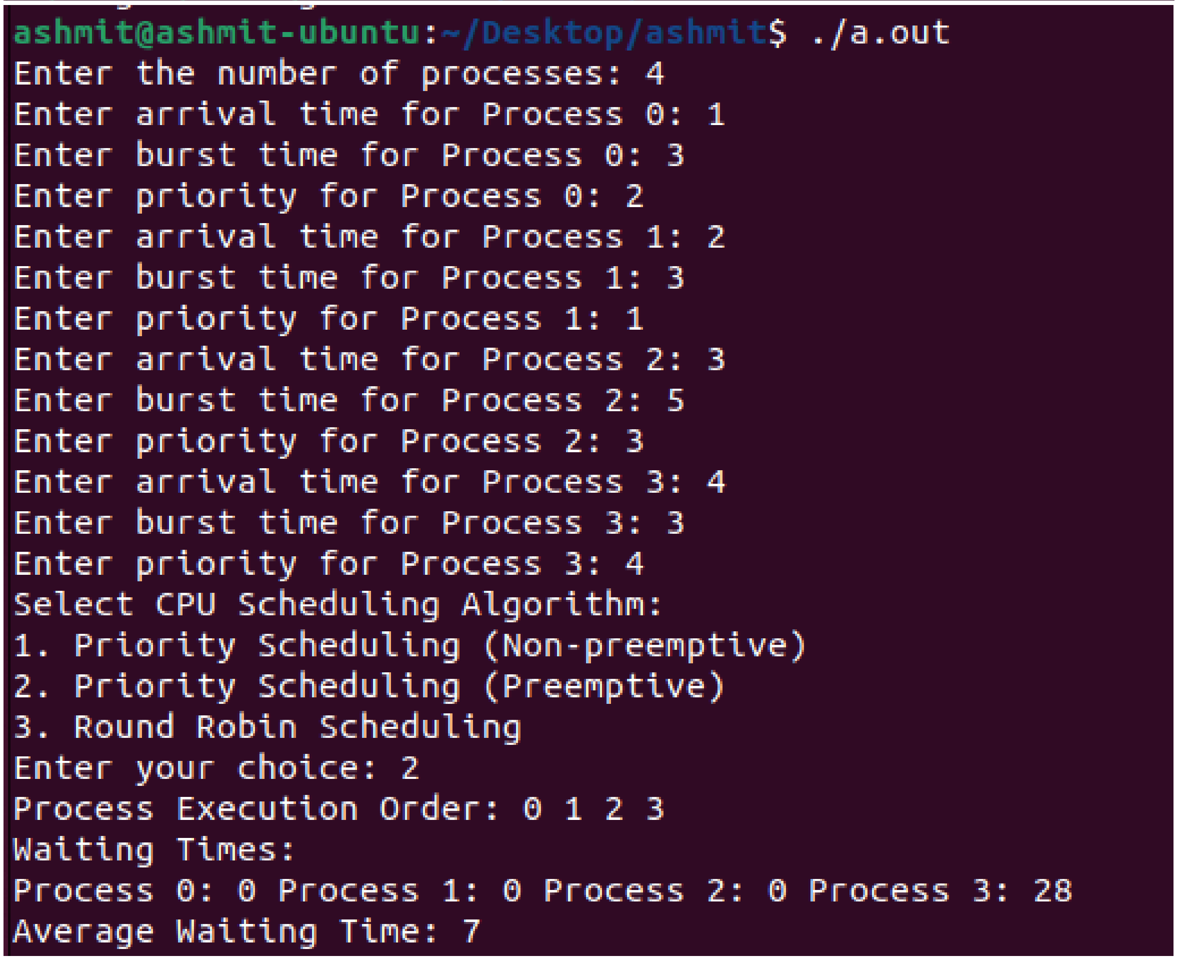
```
    int completed = 0;

    while (completed < n) {
        for (int i = 0; i < n; i++) {
            if (processes[i].arrivalTime <= currentTime && remainingTime[i] > 0) {
                int executeTime = min(quantum, remainingTime[i]);
                currentTime += executeTime;
                remainingTime[i] -= executeTime;
                q.push(processes[i]);

                if (remainingTime[i] == 0) {
                    completed += 1;
                    waitingTime[processes[i].id] = currentTime - processes[i].arrivalTime - processes[i].burstTime;
                }
            }
        }

        if (!q.empty()) {
            Process current = q.front();
            q.pop();
            q.push(current);
        }
        else {
            currentTime += 1;
        }
    }

    double avgWaitingTime = 0.0;
    for (int i = 0; i < n; i++) {
        avgWaitingTime += waitingTime[i];
    }

    avgWaitingTime /= n;

    cout << "Process Execution Order: ";
    for (int i = 0; i < n; i++) {
        cout << processes[i].id << " ";
    }

    cout << "\nWaiting Times:\n";
    for (int i = 0; i < n; i++) {
        cout << "Process " << processes[i].id << ": " << waitingTime[i] << " ";
    }

    cout << "\nAverage Waiting Time: " << avgWaitingTime << endl;
```



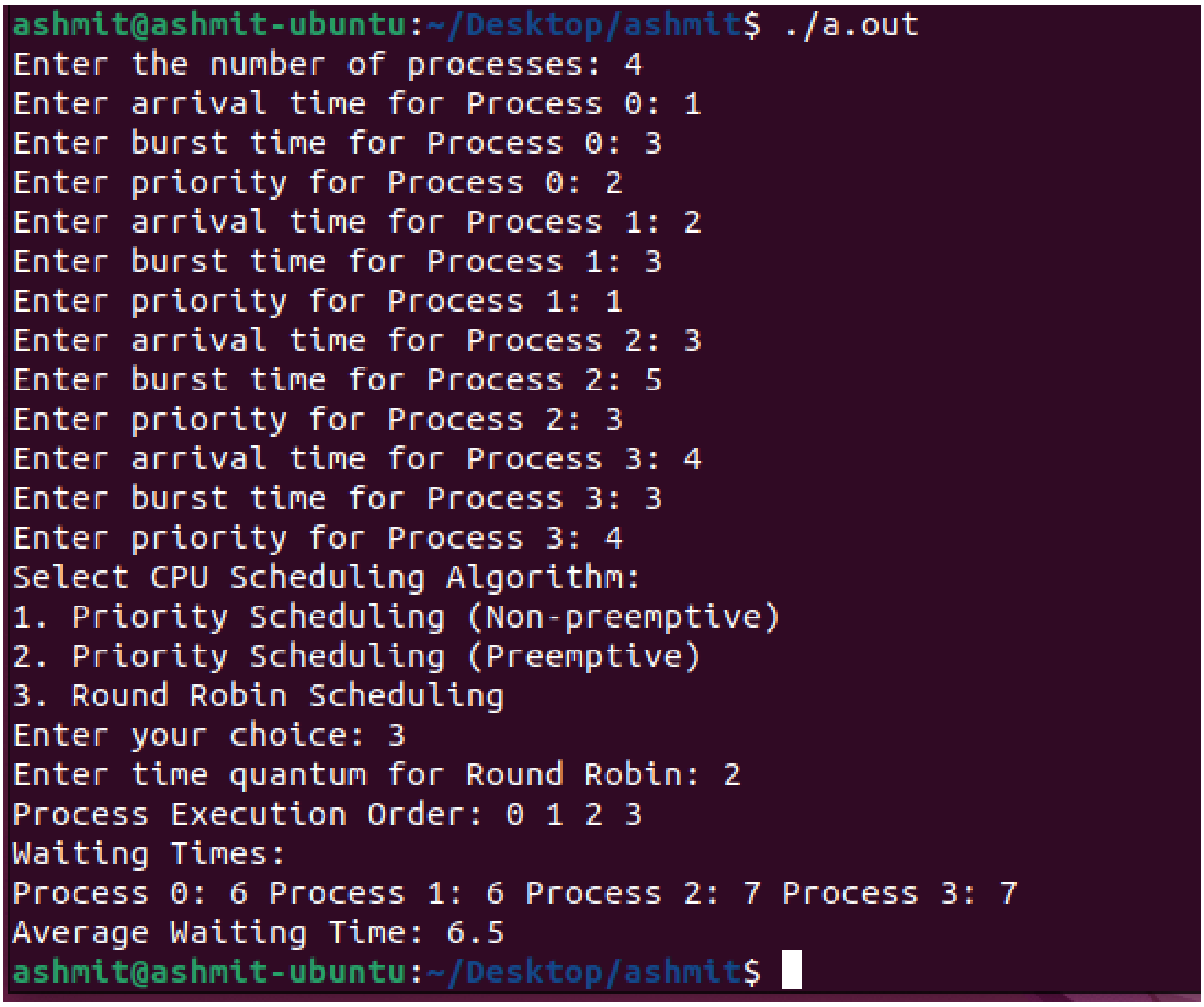
```
int main() {
    int n, algorithm;
    cout << "Enter the number of processes: ";
    cin >> n;

    vector<Process> processes(n);

    for (int i = 0; i < n; i++) {
        processes[i].id = i;
        cout << "Enter arrival time for Process " << i << ": ";
        cin >> processes[i].arrivalTime;
        cout << "Enter burst time for Process " << i << ": ";
        cin >> processes[i].burstTime;
        cout << "Enter priority for Process " << i << ": ";
        cin >> processes[i].priority;
        processes[i].remainingTime = processes[i].burstTime;
    }

    cout << "Select CPU Scheduling Algorithm:\n";
    cout << "1. Priority Scheduling (Non-preemptive)\n";
    cout << "2. Priority Scheduling (Preemptive)\n";
    cout << "3. Round Robin Scheduling\n";
    cout << "Enter your choice: ";
    cin >> algorithm;

    switch (algorithm) {
        case 1:
            priorityNonPreemptive(processes);
            break;
        case 2:
            priorityPreemptive(processes);
            break;
        case 3:
            int quantum;
            cout << "Enter time quantum for Round Robin: ";
            cin >> quantum;
            roundRobin(processes, quantum);
            break;
        default:
            cout << "Invalid choice\n";
    }

    return 0;
}
```



```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ g++ assign6.cpp
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the number of processes: 4
Enter arrival time for Process 0: 1
Enter burst time for Process 0: 3
Enter priority for Process 0: 2
Enter arrival time for Process 1: 2
Enter burst time for Process 1: 3
Enter priority for Process 1: 1
Enter arrival time for Process 2: 3
Enter burst time for Process 2: 5
Enter priority for Process 2: 3
Enter arrival time for Process 3: 4
Enter burst time for Process 3: 3
Enter priority for Process 3: 4
Select CPU Scheduling Algorithm:
1. Priority Scheduling (Non-preemptive)
2. Priority Scheduling (Preemptive)
3. Round Robin Scheduling
Enter your choice: 1
Process Execution Order: 1 0 2 3
Waiting Times:
Process 1: 0 Process 0: 3 Process 2: 6 Process 3: 11
Average Waiting Time: 5
```



```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the number of processes: 4
Enter arrival time for Process 0: 1
Enter burst time for Process 0: 3
Enter priority for Process 0: 2
Enter arrival time for Process 1: 2
Enter burst time for Process 1: 3
Enter priority for Process 1: 1
Enter arrival time for Process 2: 3
Enter burst time for Process 2: 5
Enter priority for Process 2: 3
Enter arrival time for Process 3: 4
Enter burst time for Process 3: 3
Enter priority for Process 3: 4
Select CPU Scheduling Algorithm:
1. Priority Scheduling (Non-preemptive)
2. Priority Scheduling (Preemptive)
3. Round Robin Scheduling
Enter your choice: 2
Process Execution Order: 0 1 2 3
Waiting Times:
Process 0: 0 Process 1: 0 Process 2: 0 Process 3: 28
Average Waiting Time: 7
```



```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the number of processes: 4
Enter arrival time for Process 0: 1
Enter burst time for Process 0: 3
Enter priority for Process 0: 2
Enter arrival time for Process 1: 2
Enter burst time for Process 1: 3
Enter priority for Process 1: 1
Enter arrival time for Process 2: 3
Enter burst time for Process 2: 5
Enter priority for Process 2: 3
Enter arrival time for Process 3: 4
Enter burst time for Process 3: 3
Enter priority for Process 3: 4
Select CPU Scheduling Algorithm:
1. Priority Scheduling (Non-preemptive)
2. Priority Scheduling (Preemptive)
3. Round Robin Scheduling
Enter your choice: 3
Enter time quantum for Round Robin: 2
Process Execution Order: 0 1 2 3
Waiting Times:
Process 0: 6 Process 1: 6 Process 2: 7 Process 3: 7
Average Waiting Time: 6.5
ashmit@ashmit-ubuntu:~/Desktop/ashmit$
```