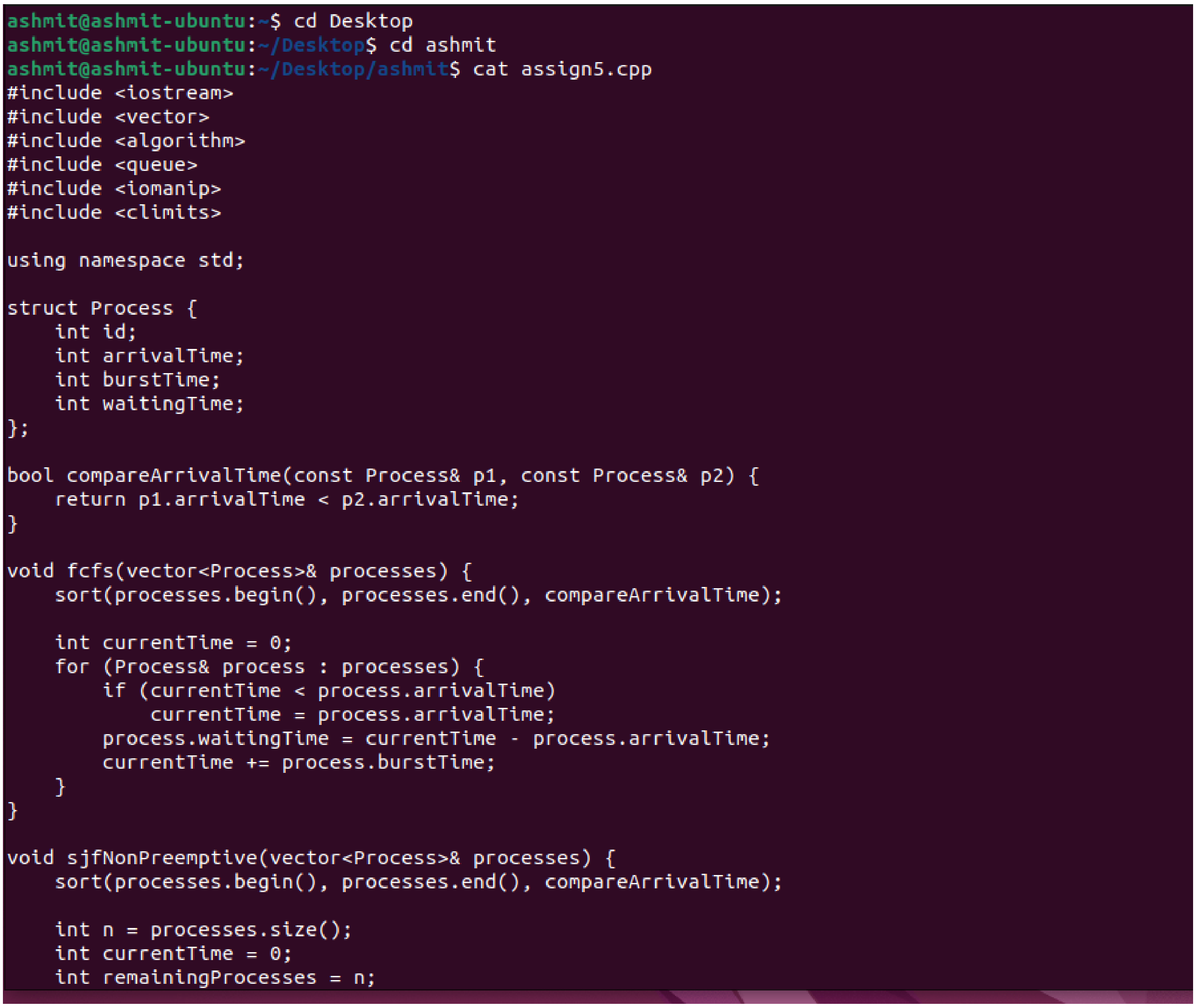
Name: Ashmit Thawait

Roll No: 102203790

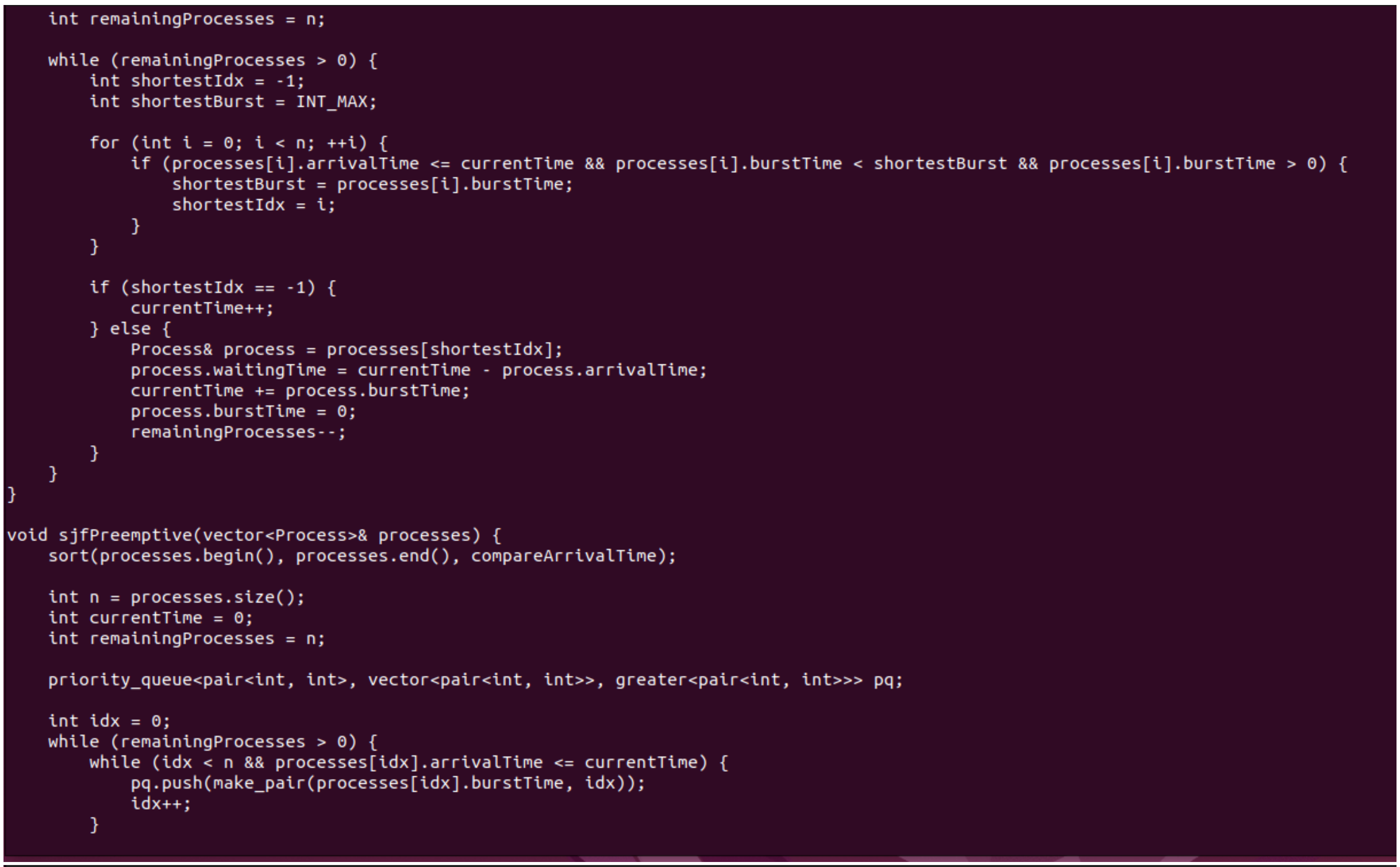
2CO-17

OS Assignment 5

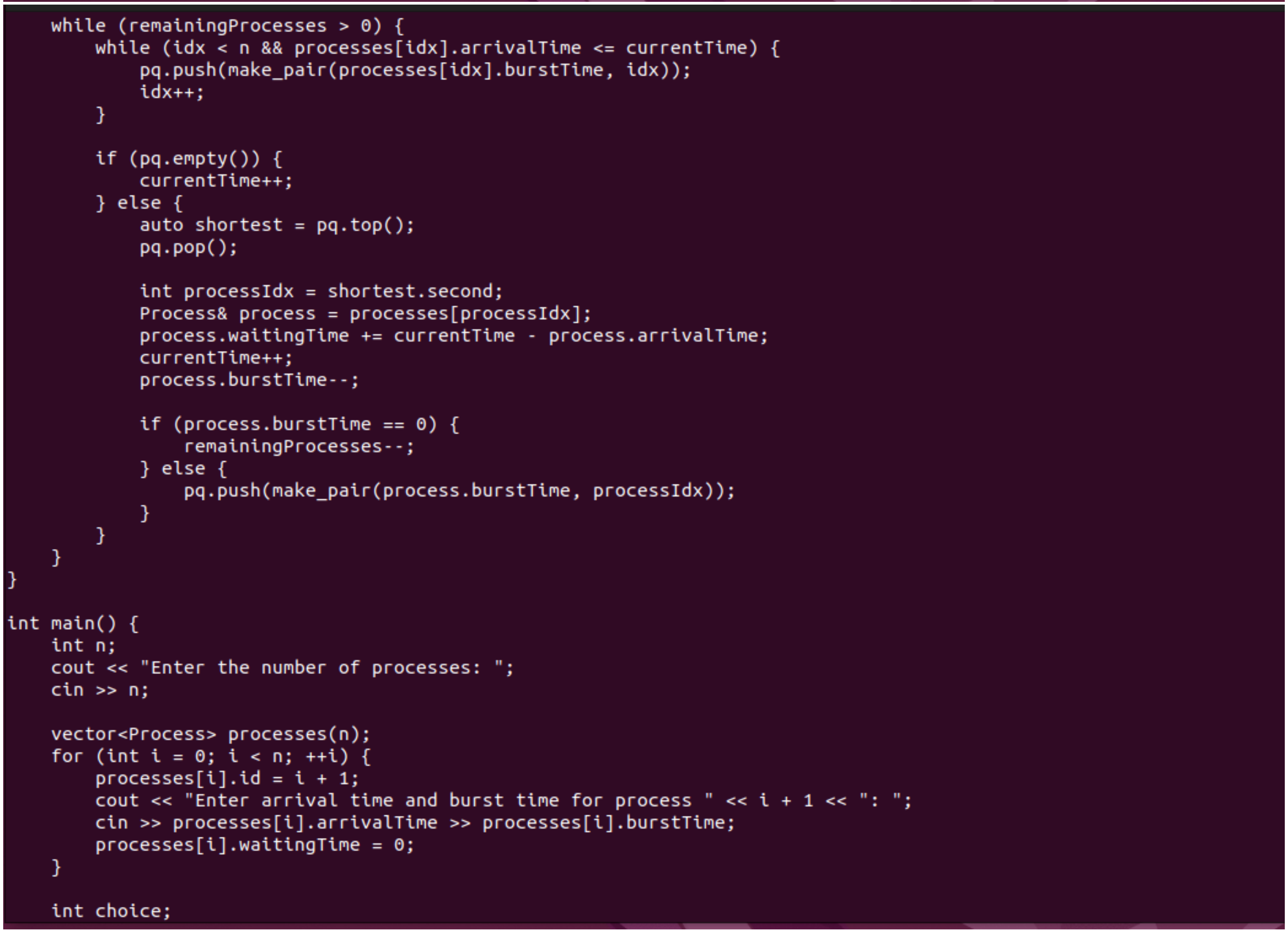
Q. Write a program using C/C++/Java to simulate the FCFS, SJF (pre-emptive as well as non pre-emptive approach). The scenario is: user may input n processes with respective CPU burst time and arrival time. System will ask the user to select the type of algorithm from the list mentioned above. System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



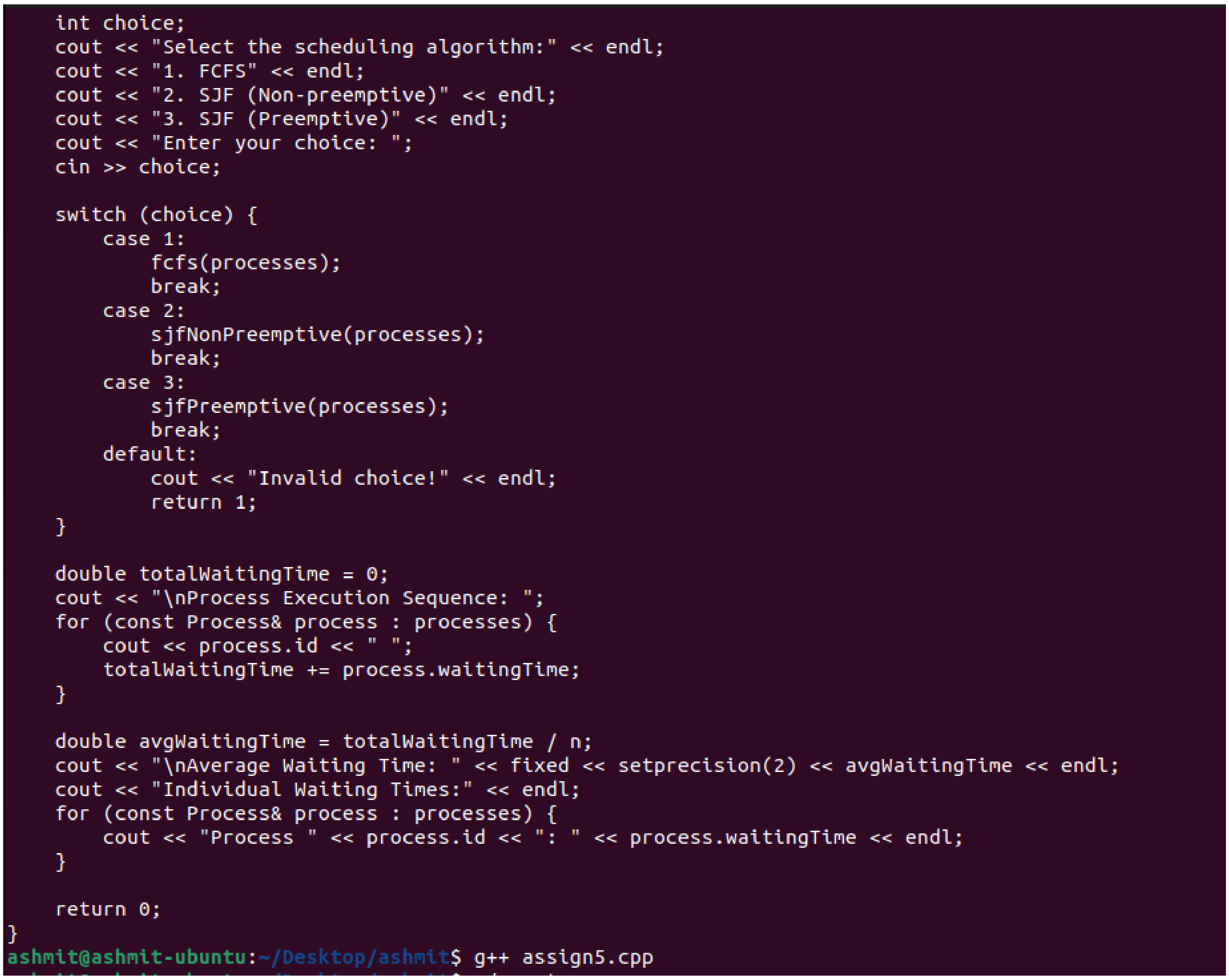
```
ashmit@ashmit-ubuntu:~$ cd Desktop
ashmit@ashmit-ubuntu:~/Desktop$ cd ashmit
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ cat assign5.cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
#include <iomanip>
#include <climits>

using namespace std;

struct Process {
    int id;
    int arrivalTime;
    int burstTime;
    int waitingTime;
};

bool compareArrivalTime(const Process& p1, const Process& p2) {
    return p1.arrivalTime < p2.arrivalTime;
}

void fcfs(vector<Process>& processes) {
    sort(processes.begin(), processes.end(), compareArrivalTime);

    int currentTime = 0;
    for (Process& process : processes) {
        if (currentTime < process.arrivalTime)
            currentTime = process.arrivalTime;
        process.waitingTime = currentTime - process.arrivalTime;
        currentTime += process.burstTime;
    }
}

void sjfNonPreemptive(vector<Process>& processes) {
    sort(processes.begin(), processes.end(), compareArrivalTime);

    int n = processes.size();
    int currentTime = 0;
    int remainingProcesses = n;
```



```
    int remainingProcesses = n;

    while (remainingProcesses > 0) {
        int shortestIdx = -1;
        int shortestBurst = INT_MAX;

        for (int i = 0; i < n; ++i) {
            if (processes[i].arrivalTime <= currentTime && processes[i].burstTime < shortestBurst && processes[i].burstTime > 0) {
                shortestBurst = processes[i].burstTime;
                shortestIdx = i;
            }
        }

        if (shortestIdx == -1) {
            currentTime++;
        } else {
            Process& process = processes[shortestIdx];
            process.waitingTime = currentTime - process.arrivalTime;
            currentTime += process.burstTime;
            process.burstTime = 0;
            remainingProcesses--;
        }
    }
}

void sjfPreemptive(vector<Process>& processes) {
    sort(processes.begin(), processes.end(), compareArrivalTime);

    int n = processes.size();
    int currentTime = 0;
    int remainingProcesses = n;

    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

    int idx = 0;
    while (remainingProcesses > 0) {
        while (idx < n && processes[idx].arrivalTime <= currentTime) {
            pq.push(make_pair(processes[idx].burstTime, idx));
            idx++;
        }
```

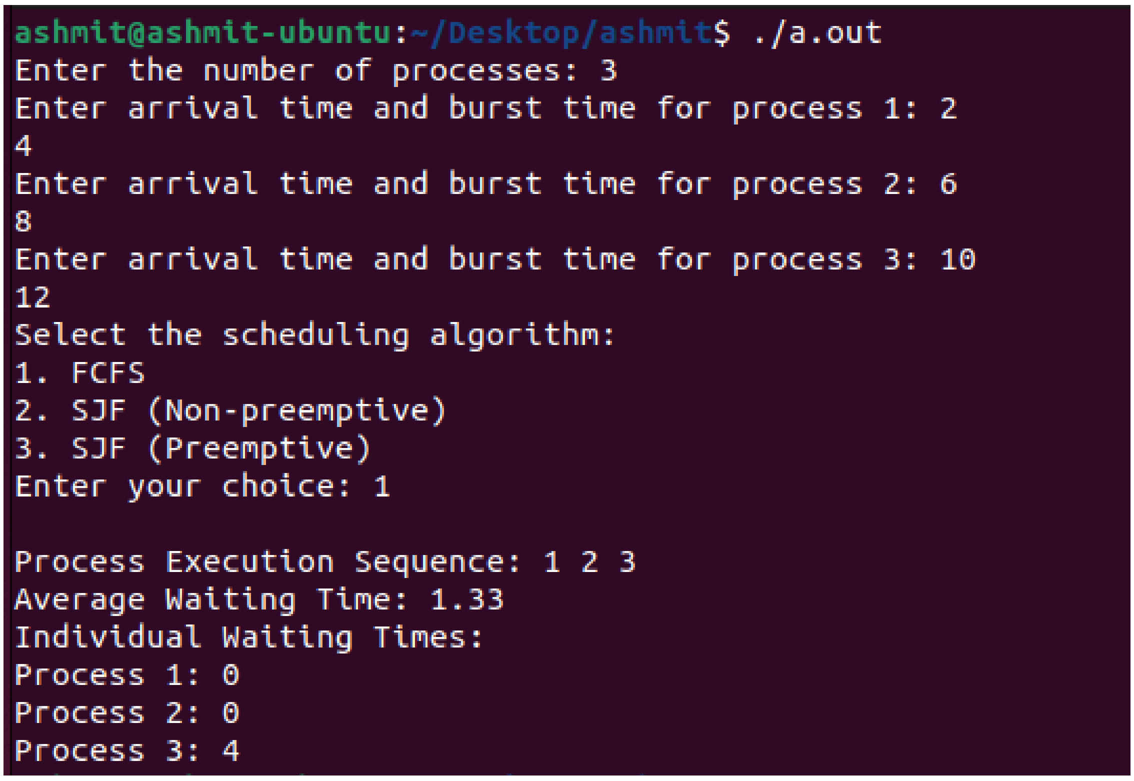


```
    while (remainingProcesses > 0) {
        while (idx < n && processes[idx].arrivalTime <= currentTime) {
            pq.push(make_pair(processes[idx].burstTime, idx));
            idx++;
        }

        if (pq.empty()) {
            currentTime++;
        } else {
            auto shortest = pq.top();
            pq.pop();

            int processIdx = shortest.second;
            Process& process = processes[processIdx];
            process.waitingTime += currentTime - process.arrivalTime;
            currentTime++;
            process.burstTime--;

            if (process.burstTime == 0) {
                remainingProcesses--;
            } else {
                pq.push(make_pair(process.burstTime, processIdx));
            }
        }
    }
}

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

    vector<Process> processes(n);
    for (int i = 0; i < n; ++i) {
        processes[i].id = i + 1;
        cout << "Enter arrival time and burst time for process " << i + 1 << ": ";
        cin >> processes[i].arrivalTime >> processes[i].burstTime;
        processes[i].waitingTime = 0;
    }

    int choice;
```



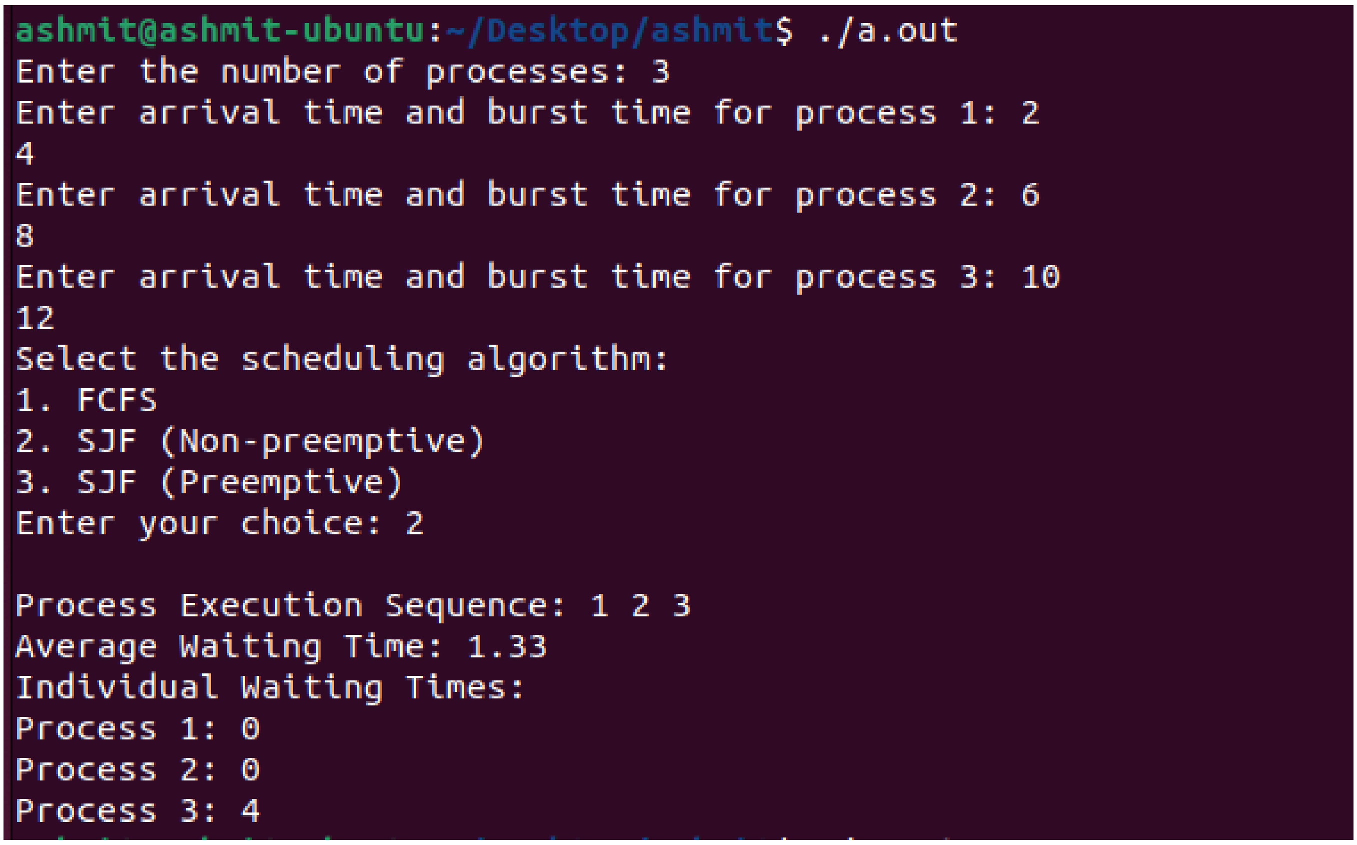
```
    int choice;
    cout << "Select the scheduling algorithm:" << endl;
    cout << "1. FCFS" << endl;
    cout << "2. SJF (Non-preemptive)" << endl;
    cout << "3. SJF (Preemptive)" << endl;
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            fcfs(processes);
            break;
        case 2:
            sjfNonPreemptive(processes);
            break;
        case 3:
            sjfPreemptive(processes);
            break;
        default:
            cout << "Invalid choice!" << endl;
            return 1;
    }

    double totalWaitingTime = 0;
    cout << "\nProcess Execution Sequence: ";
    for (const Process& process : processes) {
        cout << process.id << " ";
        totalWaitingTime += process.waitingTime;
    }

    double avgWaitingTime = totalWaitingTime / n;
    cout << "\nAverage Waiting Time: " << fixed << setprecision(2) << avgWaitingTime << endl;
    cout << "Individual Waiting Times:" << endl;
    for (const Process& process : processes) {
        cout << "Process " << process.id << ": " << process.waitingTime << endl;
    }

    return 0;
}
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ g++ assign5.cpp
```

FCFS -



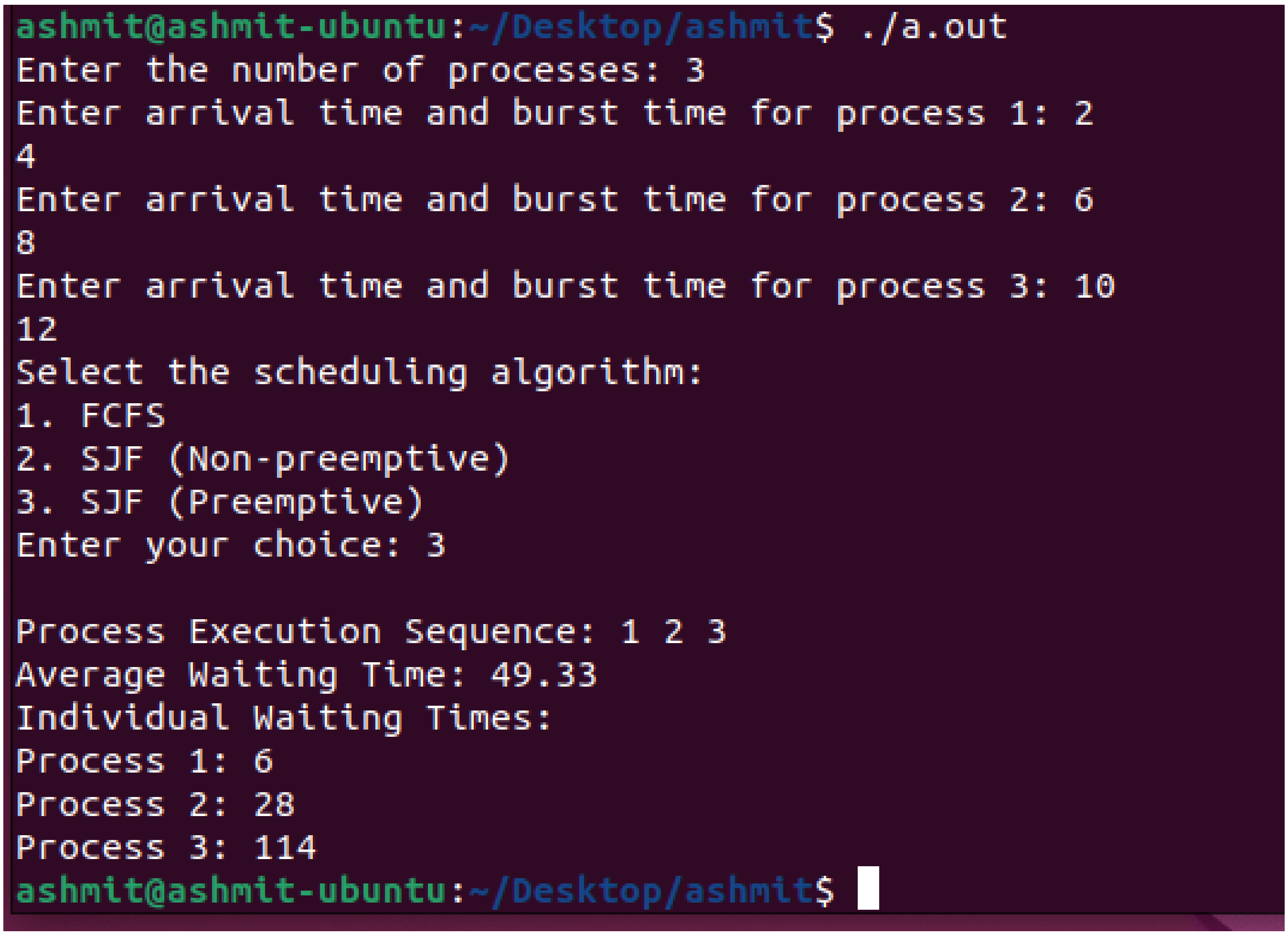
```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the number of processes: 3
Enter arrival time and burst time for process 1: 2
4
Enter arrival time and burst time for process 2: 6
8
Enter arrival time and burst time for process 3: 10
12
Select the scheduling algorithm:
1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
Enter your choice: 1

Process Execution Sequence: 1 2 3
Average Waiting Time: 1.33
Individual Waiting Times:
Process 1: 0
Process 2: 0
Process 3: 4
```

SJF (Non-preemptive) -



```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the number of processes: 3
Enter arrival time and burst time for process 1: 2
4
Enter arrival time and burst time for process 2: 6
8
Enter arrival time and burst time for process 3: 10
12
Select the scheduling algorithm:
1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
Enter your choice: 2

Process Execution Sequence: 1 2 3
Average Waiting Time: 1.33
Individual Waiting Times:
Process 1: 0
Process 2: 0
Process 3: 4
```

SJF (Preemptive) -



```
ashmit@ashmit-ubuntu:~/Desktop/ashmit$ ./a.out
Enter the number of processes: 3
Enter arrival time and burst time for process 1: 2
4
Enter arrival time and burst time for process 2: 6
8
Enter arrival time and burst time for process 3: 10
12
Select the scheduling algorithm:
1. FCFS
2. SJF (Non-preemptive)
3. SJF (Preemptive)
Enter your choice: 3

Process Execution Sequence: 1 2 3
Average Waiting Time: 49.33
Individual Waiting Times:
Process 1: 6
Process 2: 28
Process 3: 114
ashmit@ashmit-ubuntu:~/Desktop/ashmit$
```