King Saud University
College of Computer and Information Sciences
CSC227 Operating Systems

جامعة الملك سعود
King Saud University

كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات

Group #3

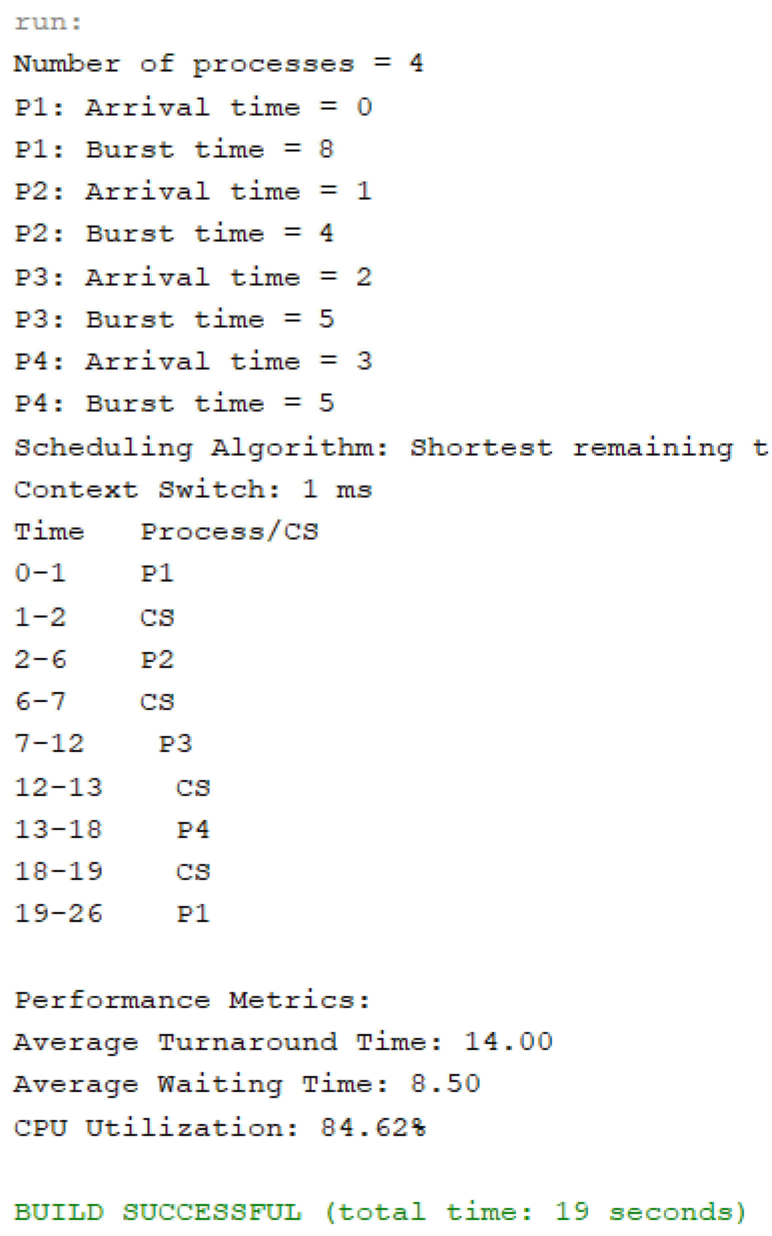
Programming Assignment 1 – Process Scheduling Simulator

Name	**ID**
Lamees Alghamdi	**444201177**
Ghadeer Alnuwaysir	444200420
Maha Alruwais	444200749
Norah Almadhi	444200890
Sara Aloqiel	444203016

Work distribution	
Name	**Role**
Ghadeer Alnuwaysir	Work divided equally
Maha Alruwais	Work divided equally
Norah Almadhi	Work divided equally
Lamees Alghamdi	Work divided equally
Sara Aloqiel	Work divided equally

**It's important to note that the work was done in person/through online meetings, to ensure equal team contribution.*

Output:



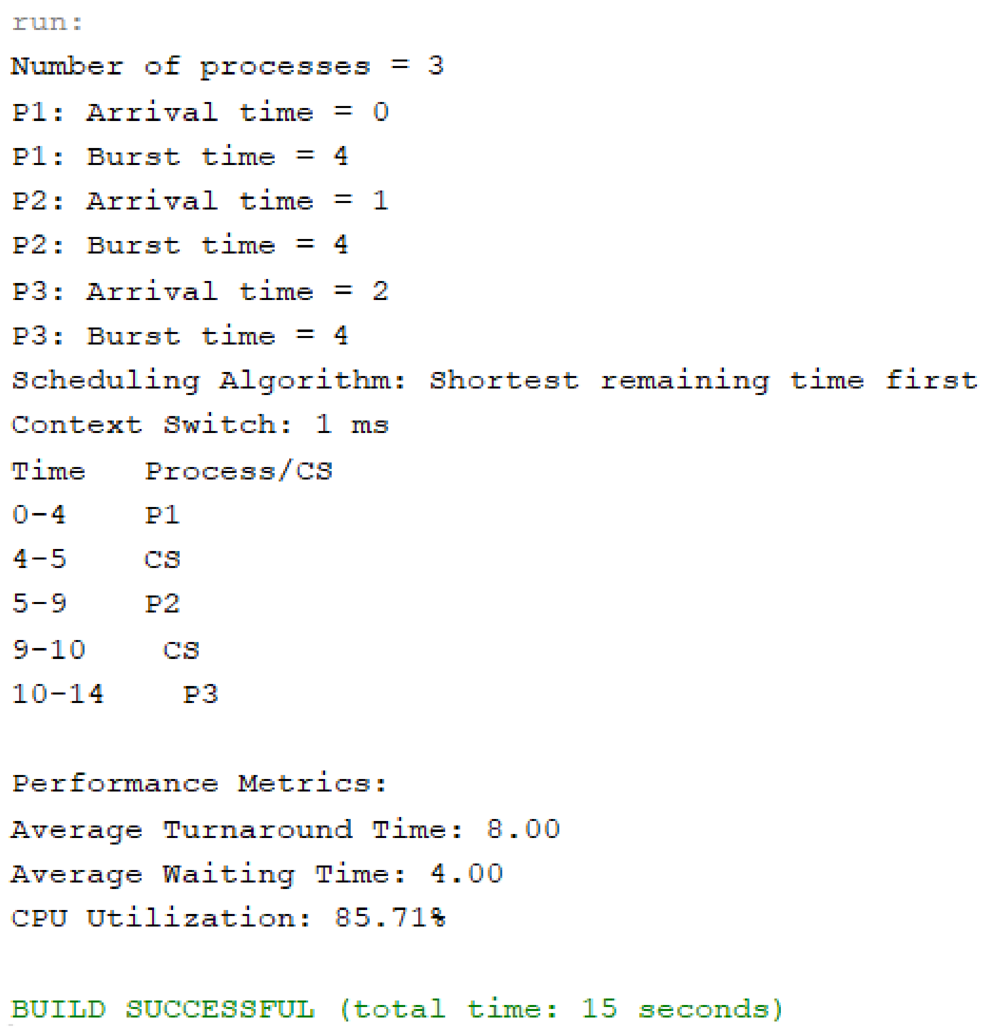
```
run:
Number of processes = 4
P1: Arrival time = 0
P1: Burst time = 8
P2: Arrival time = 1
P2: Burst time = 4
P3: Arrival time = 2
P3: Burst time = 5
P4: Arrival time = 3
P4: Burst time = 5
Scheduling Algorithm: Shortest remaining time first
Context Switch: 1 ms
Time    Process/CS
0-1     P1
1-2     CS
2-6     P2
6-7     CS
7-12     P3
12-13     CS
13-18     P4
18-19     CS
19-26     P1

Performance Metrics:
Average Turnaround Time: 14.00
Average Waiting Time: 8.50
CPU Utilization: 84.62%

BUILD SUCCESSFUL (total time: 19 seconds)
```

Additional sample runs:

1. Demonstrates FCFS condition:



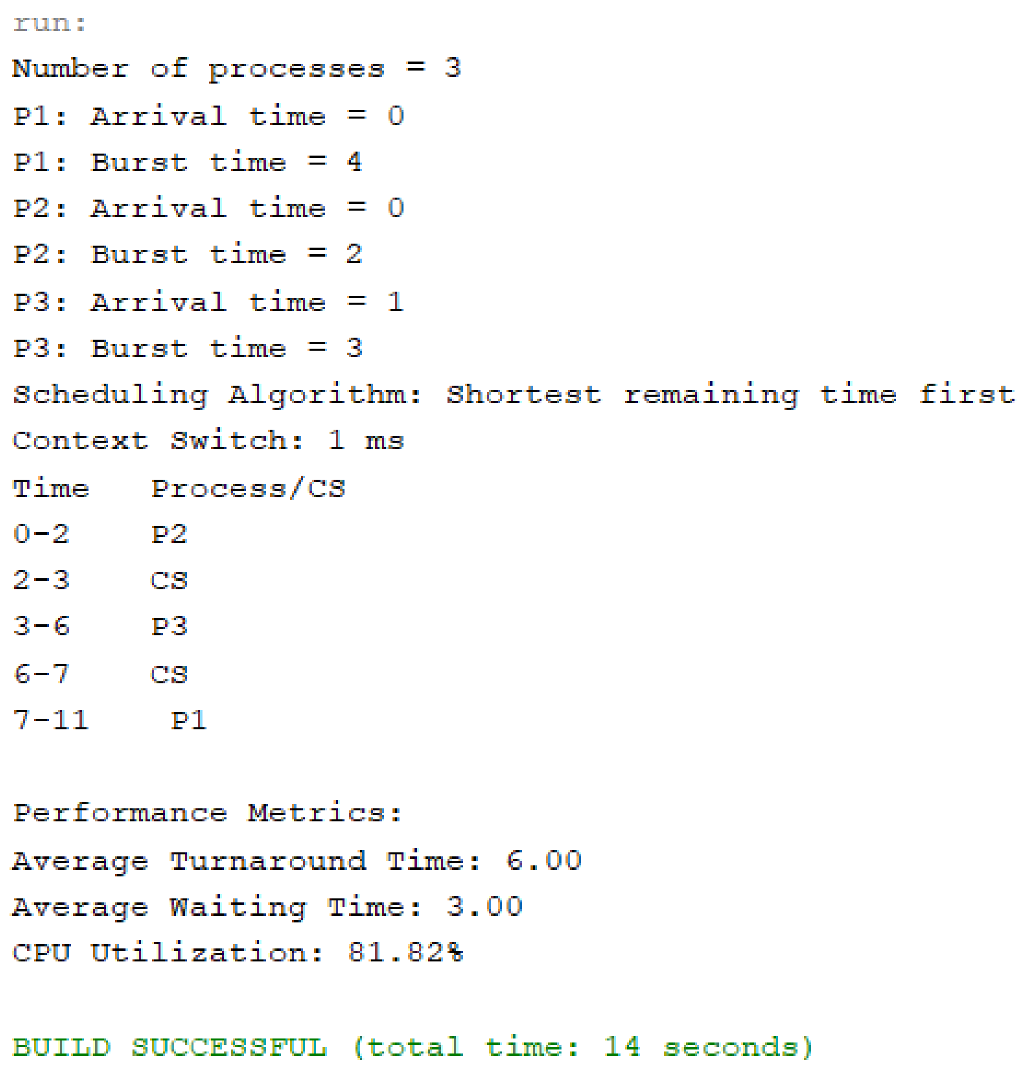
```
run:
Number of processes = 3
P1: Arrival time = 0
P1: Burst time = 4
P2: Arrival time = 1
P2: Burst time = 4
P3: Arrival time = 2
P3: Burst time = 4
Scheduling Algorithm: Shortest remaining time first
Context Switch: 1 ms
Time   Process/CS
0-4    P1
4-5    CS
5-9    P2
9-10    CS
10-14    P3

Performance Metrics:
Average Turnaround Time: 8.00
Average Waiting Time: 4.00
CPU Utilization: 85.71%

BUILD SUCCESSFUL (total time: 15 seconds)
```

This output demonstrates the FCFS condition because P1, P2, and P3 have equal burst times (4 ms). When multiple processes have the same shortest remaining time, the one that arrived first (FCFS rule) continues execution.

2. Demonstrates preemption:



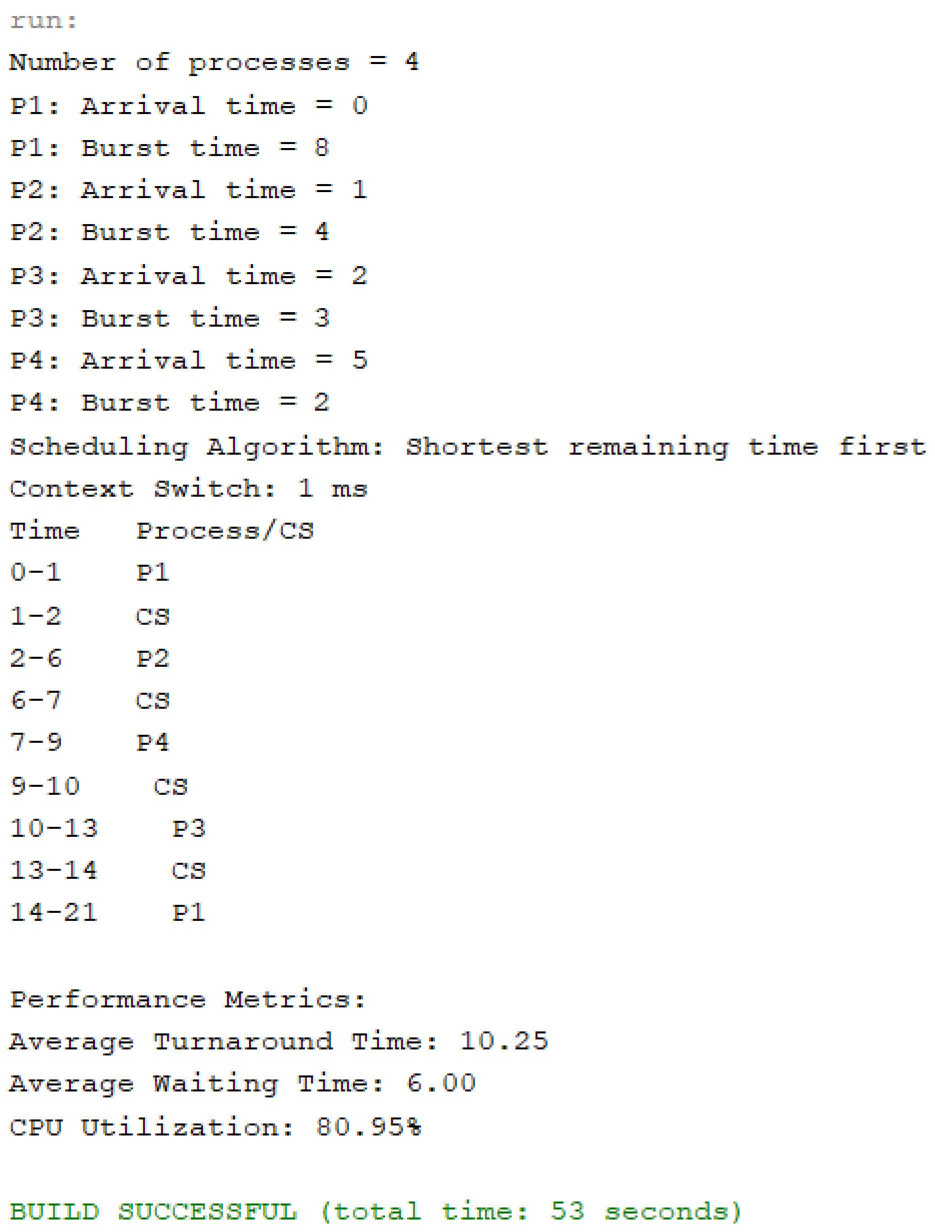
```
run:
Number of processes = 3
P1: Arrival time = 0
P1: Burst time = 4
P2: Arrival time = 0
P2: Burst time = 2
P3: Arrival time = 1
P3: Burst time = 3
Scheduling Algorithm: Shortest remaining time first
Context Switch: 1 ms
Time    Process/CS
0-2     P2
2-3     CS
3-6     P3
6-7     CS
7-11     P1

Performance Metrics:
Average Turnaround Time: 6.00
Average Waiting Time: 3.00
CPU Utilization: 81.82%

BUILD SUCCESSFUL (total time: 14 seconds)
```

This output shows preemption in SRTF scheduling, where P2 runs first, followed by P3 before P1 *due to shorter burst times*, with context switches in between.

3. Demonstrates context switch handling in SRTF:



```
run:
Number of processes = 4
P1: Arrival time = 0
P1: Burst time = 8
P2: Arrival time = 1
P2: Burst time = 4
P3: Arrival time = 2
P3: Burst time = 3
P4: Arrival time = 5
P4: Burst time = 2
Scheduling Algorithm: Shortest remaining time first
Context Switch: 1 ms
Time   Process/CS
0-1    P1
1-2    CS
2-6    P2
6-7    CS
7-9    P4
9-10    CS
10-13    P3
13-14    CS
14-21    P1

Performance Metrics:
Average Turnaround Time: 10.25
Average Waiting Time: 6.00
CPU Utilization: 80.95%

BUILD SUCCESSFUL (total time: 53 seconds)
```

This output demonstrates context switch handling in SRTF. P1 starts first, but after a context switch, P2 executes due to its shorter burst time. P4 then preempts P2, followed by P3, before P1 resumes execution. Our approach ensures that once a context switch begins, it completes before checking for new arrivals, preventing immediate preemption.