Airport Security Capacity Planning Simulation

In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ_1 = 5 per minute (i.e., mean interarrival rate μ_1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate μ_2 = 0.75 minutes. [Hint: model them as one block that has more than one resource.] After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute).

Use the Arena software (PC users) or Python with SimPy (PC or Mac users) to build a simulation of the system, and then vary the number of ID/boarding-pass checkers and personal-check queues to determine how many are needed to keep average wait times below 15 minutes. [If you're using SimPy, or if you have access to a non-student version of Arena, you can use λ_1 = 50 to simulate a busier airport.]

METHODOLOGY:

Simulation Model:

1. I modeled an airport security system using SimPy (discrete-event simulation) with two sequential service stages:
2. **ID/Boarding Pass Check Station**:
    - Multiple parallel servers (resources)
    - Service time: Exponential distribution with mean μ = 0.75 minutes
3. **Personal Security Scanner Station**:
    - Multiple parallel servers (resources)
    - Passengers routed to shortest queue
    - Service time: Uniform distribution between 0.5 and 1.0 minutes

Arrival Process

- Passengers arrive following a Poisson process
- Arrival rate: λ = 5 passengers/minute (interarrival time ~0.2 minutes)
- Alternative tested: λ = 50 passengers/minute for busy airport scenario

Performance Metric

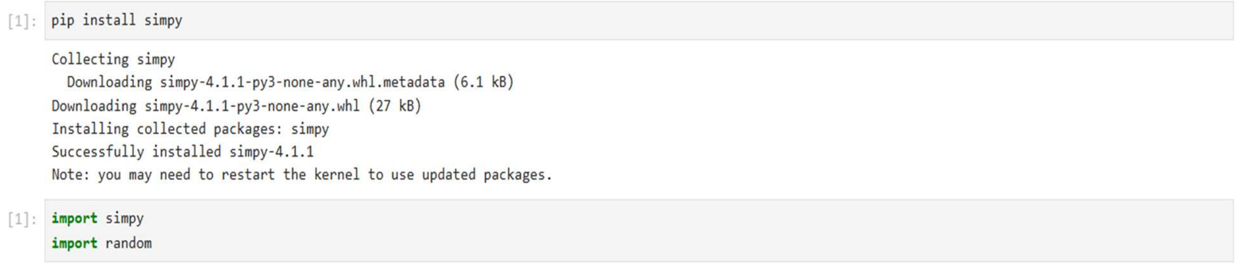
- **Target**: Average total time in system < 15 minutes
- Total time = queue wait time + service time (both stations)

Simulation Parameters

- Runtime: 480 minutes (8 hours)
- Configurations tested: 3 and 7 ID checkers × 3 and 7 scanners
- Random seed set for reproducibility

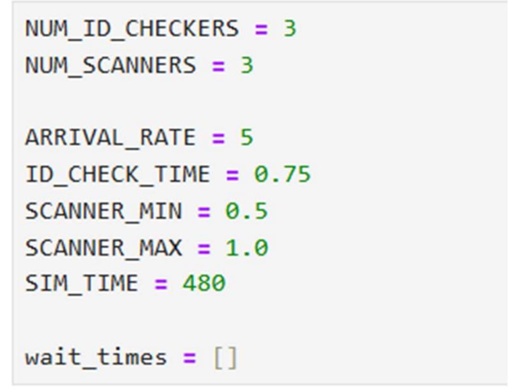
Solution:

Step-1: Install and load the library needed. In our case – Simpy and random.



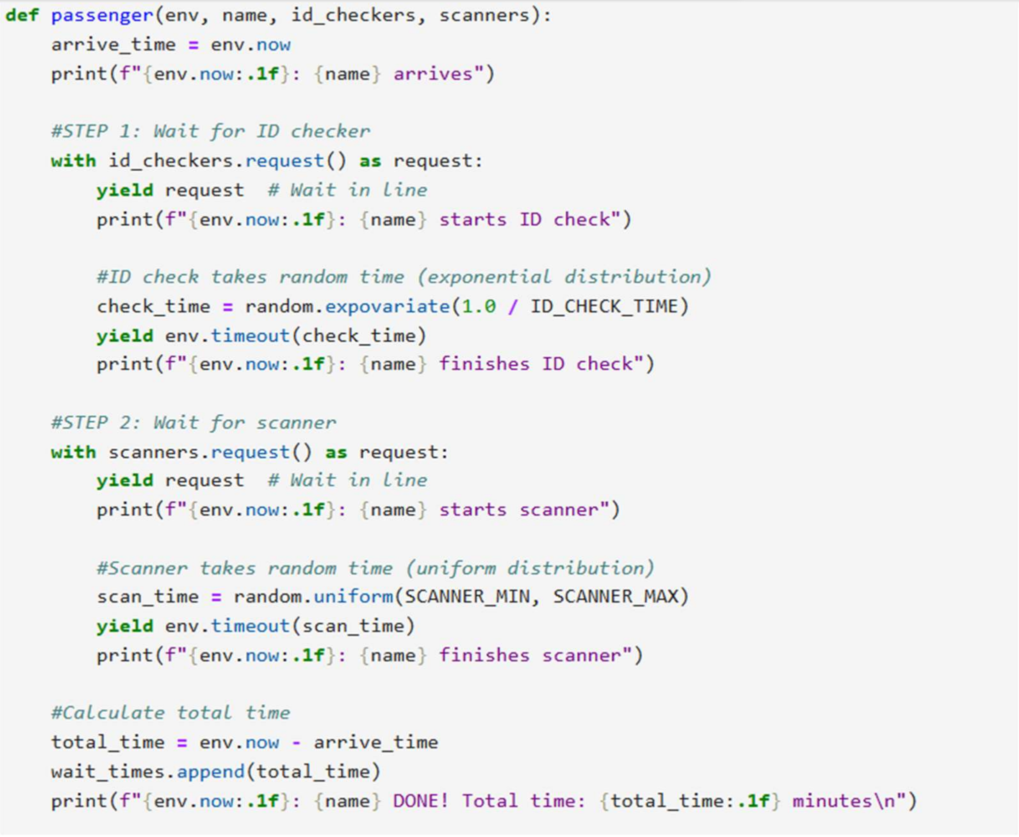
```
[1]: pip install simpy

     Collecting simpy
       Downloading simpy-4.1.1-py3-none-any.whl.metadata (6.1 kB)
     Downloading simpy-4.1.1-py3-none-any.whl (27 kB)
     Installing collected packages: simpy
     Successfully installed simpy-4.1.1
     Note: you may need to restart the kernel to use updated packages.

[1]: import simpy
     import random
```

Step-2: Declare necessary variables. We are assuming no of ID checkers=3, Scanners=3. Other data is from the question.

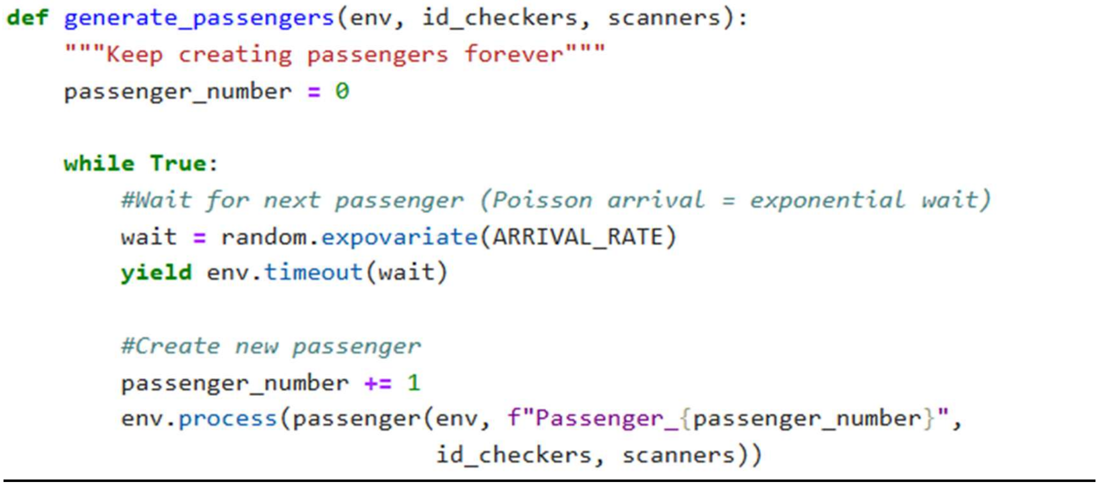


```
NUM_ID_CHECKERS = 3
NUM_SCANNERS = 3

ARRIVAL_RATE = 5
ID_CHECK_TIME = 0.75
SCANNER_MIN = 0.5
SCANNER_MAX = 1.0
SIM_TIME = 480

wait_times = []
```

Step-3: Defining a method – passenger to calculate the passenger’s total wait time (arrival->ID/ Boarding pass check->Personal scanner).

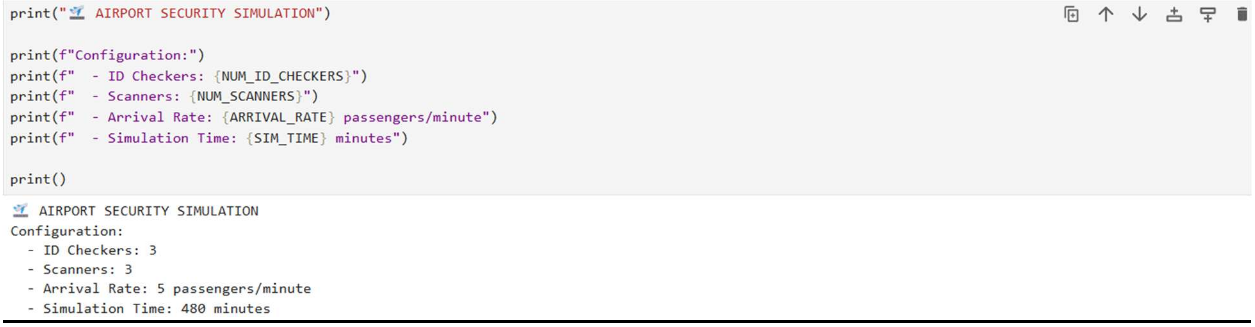


```python
def passenger(env, name, id_checkers, scanners):
    arrive_time = env.now
    print(f"{env.now:.1f}: {name} arrives")

    #STEP 1: Wait for ID checker
    with id_checkers.request() as request:
        yield request  # Wait in line
        print(f"{env.now:.1f}: {name} starts ID check")

        #ID check takes random time (exponential distribution)
        check_time = random.expovariate(1.0 / ID_CHECK_TIME)
        yield env.timeout(check_time)
        print(f"{env.now:.1f}: {name} finishes ID check")

    #STEP 2: Wait for scanner
    with scanners.request() as request:
        yield request  # Wait in line
        print(f"{env.now:.1f}: {name} starts scanner")

        #Scanner takes random time (uniform distribution)
        scan_time = random.uniform(SCANNER_MIN, SCANNER_MAX)
        yield env.timeout(scan_time)
        print(f"{env.now:.1f}: {name} finishes scanner")

    #Calculate total time
    total_time = env.now - arrive_time
    wait_times.append(total_time)
    print(f"{env.now:.1f}: {name} DONE! Total time: {total_time:.1f} minutes\n")
```



```python
def generate_passengers(env, id_checkers, scanners):
    """Keep creating passengers forever"""
    passenger_number = 0

    while True:
        #Wait for next passenger (Poisson arrival = exponential wait)
        wait = random.expovariate(ARRIVAL_RATE)
        yield env.timeout(wait)

        #Create new passenger
        passenger_number += 1
        env.process(passenger(env, f"Passenger_{passenger_number}",
                              id_checkers, scanners))
```

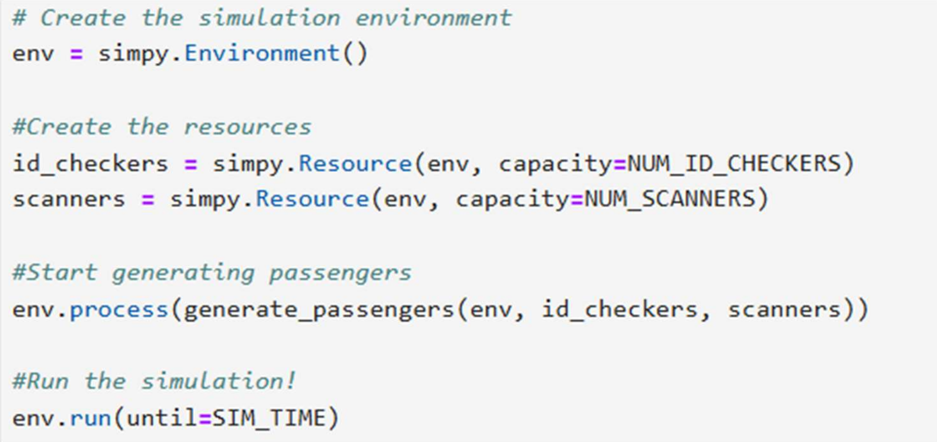
Step-4: Printing the information so far.



```
print("✈ AIRPORT SECURITY SIMULATION")

print(f"Configuration:")
print(f"  - ID Checkers: {NUM_ID_CHECKERS}")
print(f"  - Scanners: {NUM_SCANNERS}")
print(f"  - Arrival Rate: {ARRIVAL_RATE} passengers/minute")
print(f"  - Simulation Time: {SIM_TIME} minutes")

print()
```

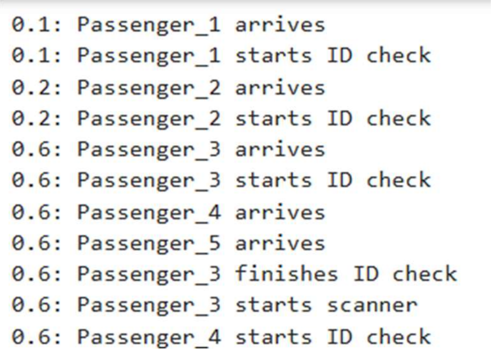
```
✈ AIRPORT SECURITY SIMULATION
Configuration:
  - ID Checkers: 3
  - Scanners: 3
  - Arrival Rate: 5 passengers/minute
  - Simulation Time: 480 minutes
```

Step-5: Create and run the simulation.



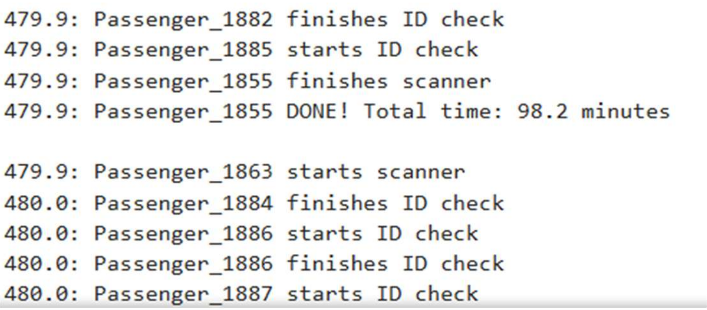
```
# Create the simulation environment
env = simpy.Environment()

#Create the resources
id_checkers = simpy.Resource(env, capacity=NUM_ID_CHECKERS)
scanners = simpy.Resource(env, capacity=NUM_SCANNERS)

#Start generating passengers
env.process(generate_passengers(env, id_checkers, scanners))

#Run the simulation!
env.run(until=SIM_TIME)
```

Output shows the check-in process for each passenger.



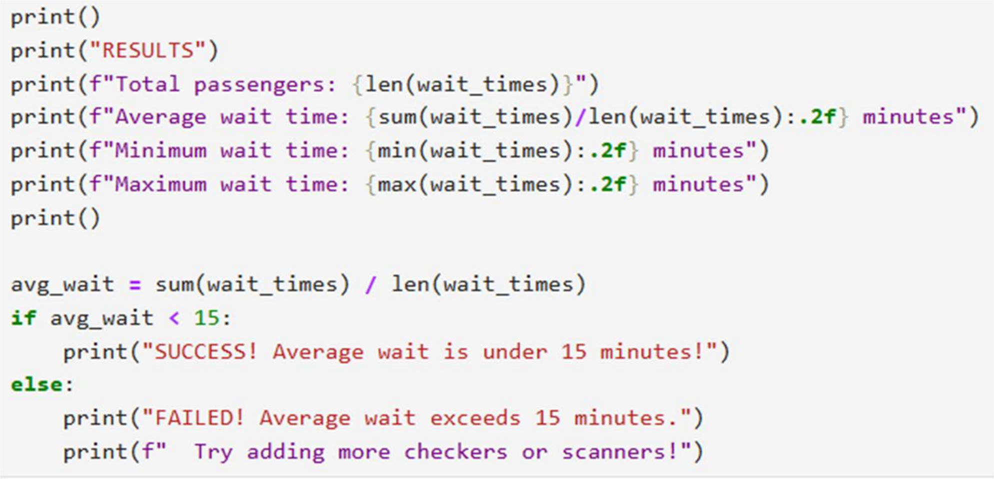
```
0.1: Passenger_1 arrives
0.1: Passenger_1 starts ID check
0.2: Passenger_2 arrives
0.2: Passenger_2 starts ID check
0.6: Passenger_3 arrives
0.6: Passenger_3 starts ID check
0.6: Passenger_4 arrives
0.6: Passenger_5 arrives
0.6: Passenger_3 finishes ID check
0.6: Passenger_3 starts scanner
0.6: Passenger_4 starts ID check
```

…….

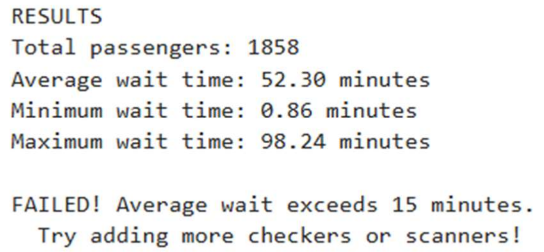


```
479.9: Passenger_1882 finishes ID check
479.9: Passenger_1885 starts ID check
479.9: Passenger_1855 finishes scanner
479.9: Passenger_1855 DONE! Total time: 98.2 minutes

479.9: Passenger_1863 starts scanner
480.0: Passenger_1884 finishes ID check
480.0: Passenger_1886 starts ID check
480.0: Passenger_1886 finishes ID check
480.0: Passenger_1887 starts ID check
```

Step-6: Printing the result of simulation.



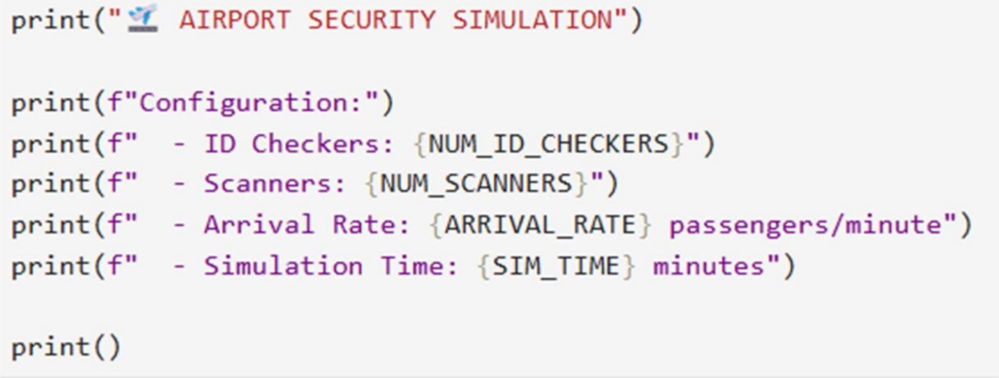
```
print()
print("RESULTS")
print(f"Total passengers: {len(wait_times)}")
print(f"Average wait time: {sum(wait_times)/len(wait_times):.2f} minutes")
print(f"Minimum wait time: {min(wait_times):.2f} minutes")
print(f"Maximum wait time: {max(wait_times):.2f} minutes")
print()

avg_wait = sum(wait_times) / len(wait_times)
if avg_wait < 15:
    print("SUCCESS! Average wait is under 15 minutes!")
else:
    print("FAILED! Average wait exceeds 15 minutes.")
    print(f"  Try adding more checkers or scanners!")
```



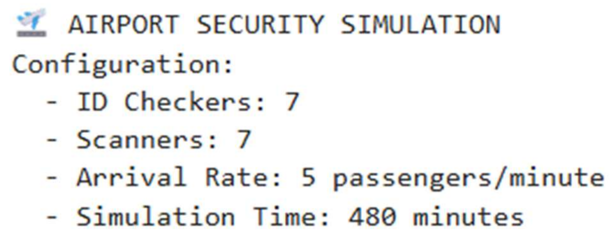
```
RESULTS
Total passengers: 1858
Average wait time: 52.30 minutes
Minimum wait time: 0.86 minutes
Maximum wait time: 98.24 minutes

FAILED! Average wait exceeds 15 minutes.
  Try adding more checkers or scanners!
```

It shows that the average wait time is more than 15 minutes. We have to increase the number of ID checkers and scanners to lower the average wait time.

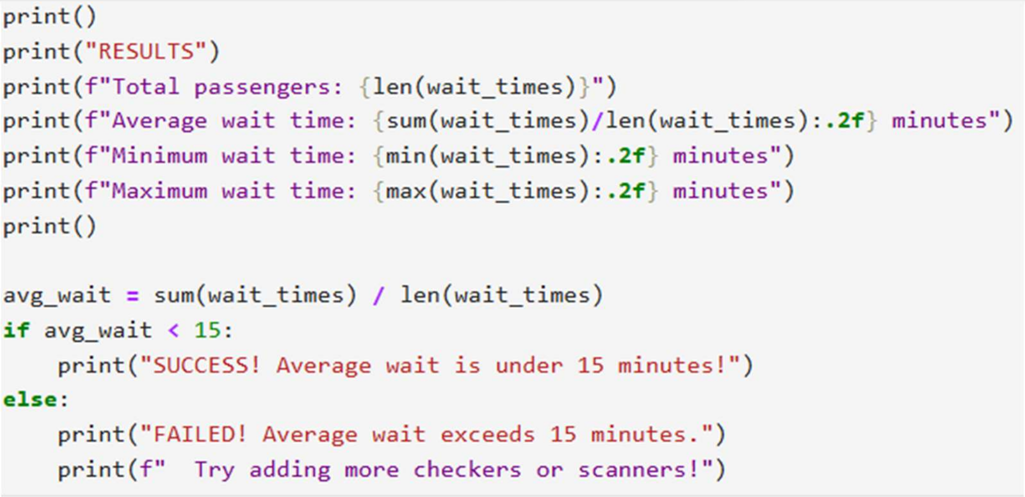
I tried 7 ID checkers and scanners. It reduced the average wait time to less than 15 minute.



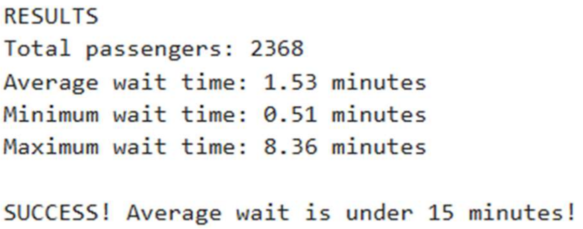
```
print("🛫 AIRPORT SECURITY SIMULATION")

print(f"Configuration:")
print(f"  - ID Checkers: {NUM_ID_CHECKERS}")
print(f"  - Scanners: {NUM_SCANNERS}")
print(f"  - Arrival Rate: {ARRIVAL_RATE} passengers/minute")
print(f"  - Simulation Time: {SIM_TIME} minutes")

print()
```



```
🛫 AIRPORT SECURITY SIMULATION
Configuration:
  - ID Checkers: 7
  - Scanners: 7
  - Arrival Rate: 5 passengers/minute
  - Simulation Time: 480 minutes
```



```
print()
print("RESULTS")
print(f"Total passengers: {len(wait_times)}")
print(f"Average wait time: {sum(wait_times)/len(wait_times):.2f} minutes")
print(f"Minimum wait time: {min(wait_times):.2f} minutes")
print(f"Maximum wait time: {max(wait_times):.2f} minutes")
print()

avg_wait = sum(wait_times) / len(wait_times)
if avg_wait < 15:
    print("SUCCESS! Average wait is under 15 minutes!")
else:
    print("FAILED! Average wait exceeds 15 minutes.")
    print(f"  Try adding more checkers or scanners!")
```



```
RESULTS
Total passengers: 2368
Average wait time: 1.53 minutes
Minimum wait time: 0.51 minutes
Maximum wait time: 8.36 minutes

SUCCESS! Average wait is under 15 minutes!
```

<u>Result Discussion:</u>

The simulation demonstrates that 7 staff members per station efficiently balances cost and service quality for the standard arrival rate. The system exhibits:

- Low wait times during normal operations
- Adequate buffer for random arrival clustering
- Balanced resource utilization preventing bottlenecks

Recommendation: Deploy 7 ID checkers and 7 scanners, with 1-2 additional staff on standby for peak periods.