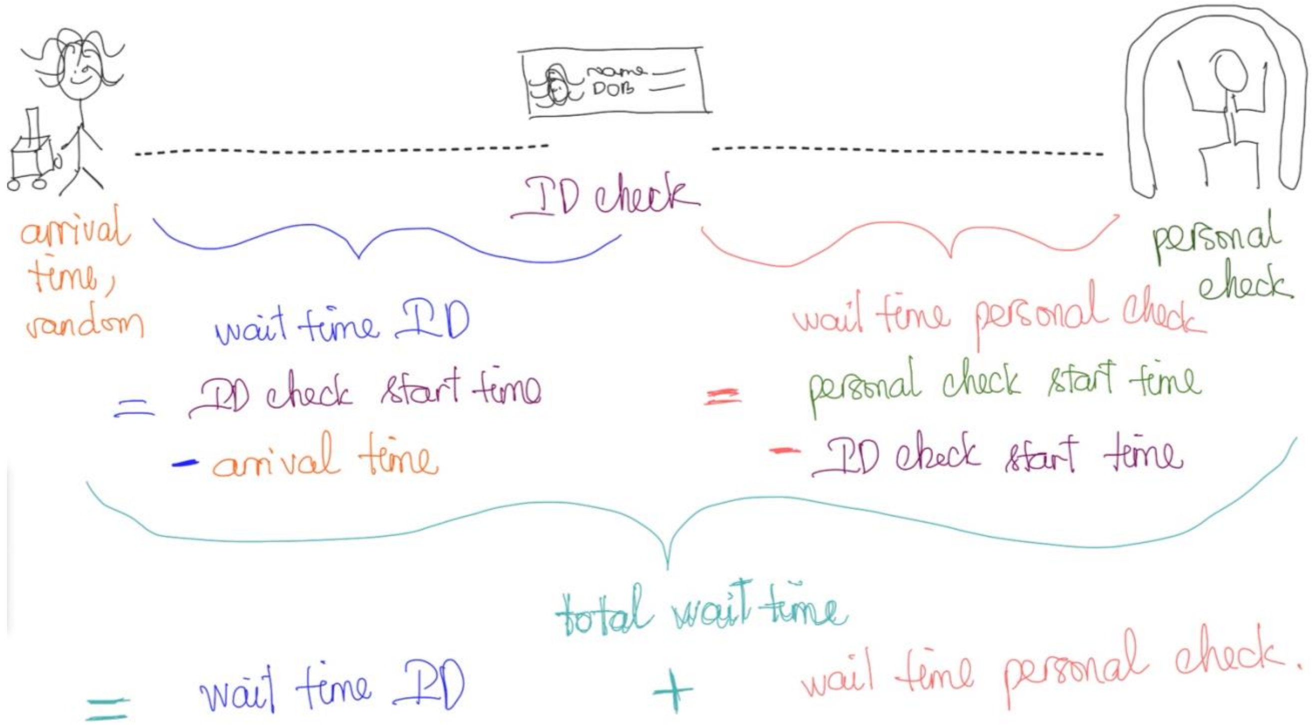
Question 13.2

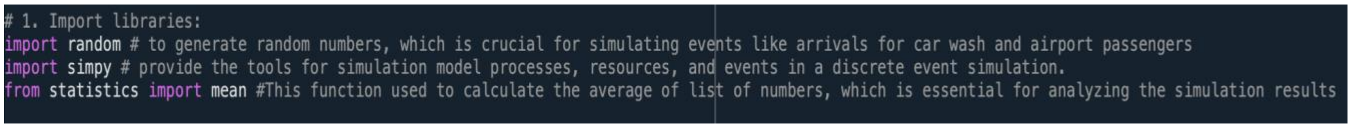
In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ_1 = 5 per minute (i.e., mean interarrival rate μ_1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential 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.]

Answer:
I am using Spyder 6 on Mac. The following steps and codes are referenced from office hour on March 10th, and AI.

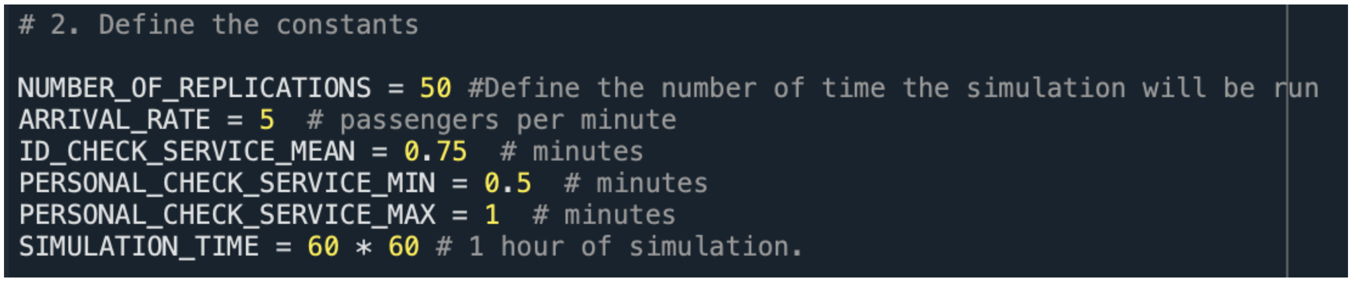


Step 1: Load the necessary libraries:



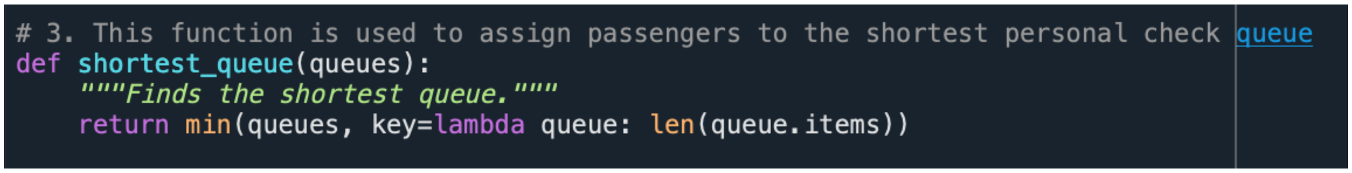
```
# 1. Import libraries:
import random # to generate random numbers, which is crucial for simulating events like arrivals for car wash and airport passengers
import simpy # provide the tools for simulation model processes, resources, and events in a discrete event simulation.
from statistics import mean #This function used to calculate the average of list of numbers, which is essential for analyzing the simulation results
```

Step 2: Define the constants that are provided in the homework:



```
# 2. Define the constants

NUMBER_OF_REPLICATIONS = 50 #Define the number of time the simulation will be run
ARRIVAL_RATE = 5  # passengers per minute
ID_CHECK_SERVICE_MEAN = 0.75  # minutes
PERSONAL_CHECK_SERVICE_MIN = 0.5  # minutes
PERSONAL_CHECK_SERVICE_MAX = 1  # minutes
SIMULATION_TIME = 60 * 60 # 1 hour of simulation.
```

Step 3: Create function is used to assign passengers to the shortest personal check queue: Since passengers choose the shortest personal check queue, this function finds the queue with the fewest people.

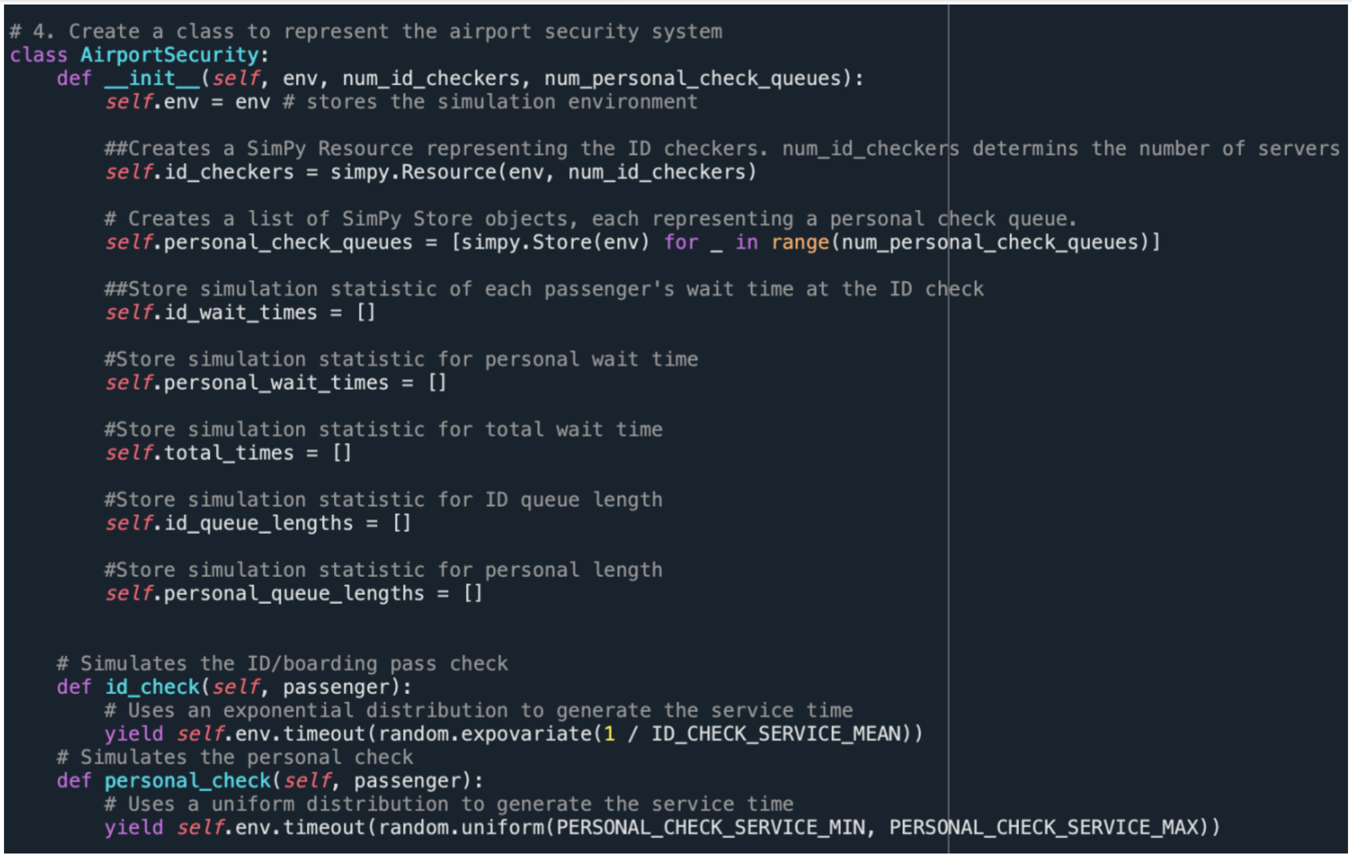


```
# 3. This function is used to assign passengers to the shortest personal check queue
def shortest_queue(queues):
    """Finds the shortest queue."""
    return min(queues, key=lambda queue: len(queue.items))
```

Step 4: Create a class to represent the airport security system:

This class models the security system with:

- ID checkers (id_checkers) → A resource with multiple ID checkers.
- Personal check queues (personal_check_queues) → A list of queues for security checks.
- Statistics storage → Lists to track waiting times and queue lengths.

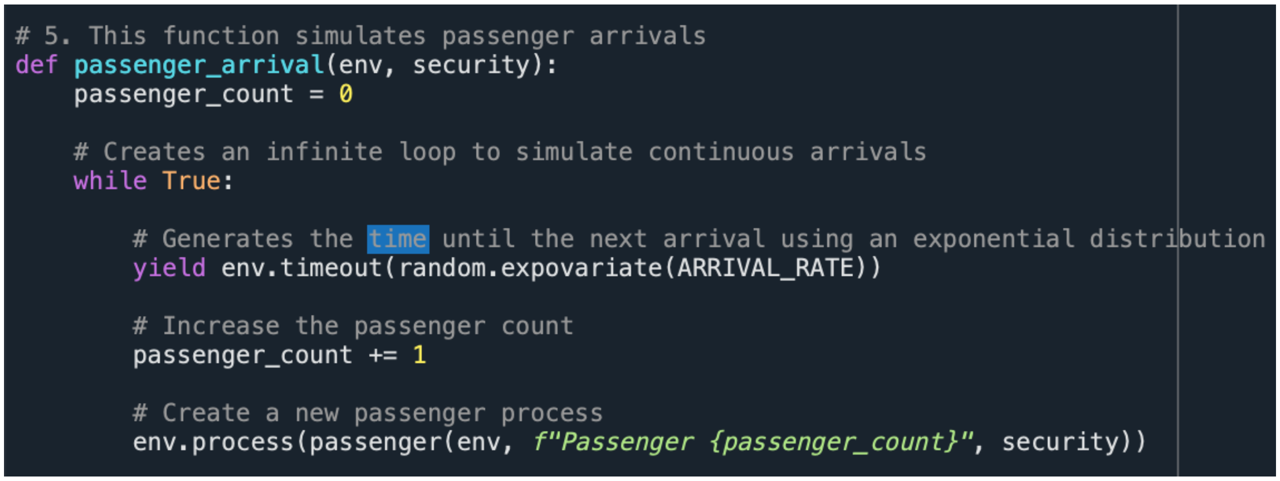


```python
# 4. Create a class to represent the airport security system
class AirportSecurity:
    def __init__(self, env, num_id_checkers, num_personal_check_queues):
        self.env = env # stores the simulation environment

        ##Creates a SimPy Resource representing the ID checkers. num_id_checkers determins the number of servers
        self.id_checkers = simpy.Resource(env, num_id_checkers)

        # Creates a list of SimPy Store objects, each representing a personal check queue.
        self.personal_check_queues = [simpy.Store(env) for _ in range(num_personal_check_queues)]

        ##Store simulation statistic of each passenger's wait time at the ID check
        self.id_wait_times = []

        #Store simulation statistic for personal wait time
        self.personal_wait_times = []

        #Store simulation statistic for total wait time
        self.total_times = []

        #Store simulation statistic for ID queue length
        self.id_queue_lengths = []

        #Store simulation statistic for personal length
        self.personal_queue_lengths = []


    # Simulates the ID/boarding pass check
    def id_check(self, passenger):
        # Uses an exponential distribution to generate the service time
        yield self.env.timeout(random.expovariate(1 / ID_CHECK_SERVICE_MEAN))
    # Simulates the personal check
    def personal_check(self, passenger):
        # Uses a uniform distribution to generate the service time
        yield self.env.timeout(random.uniform(PERSONAL_CHECK_SERVICE_MIN, PERSONAL_CHECK_SERVICE_MAX))
```

Step 5: Create a function to simulate passenger’s arrivals:

This function generates passengers:

- A new passenger arrives after a random time (following an exponential distribution).
- It creates a new passenger process and starts it.
- This continues until the simulation reaches the time limit.

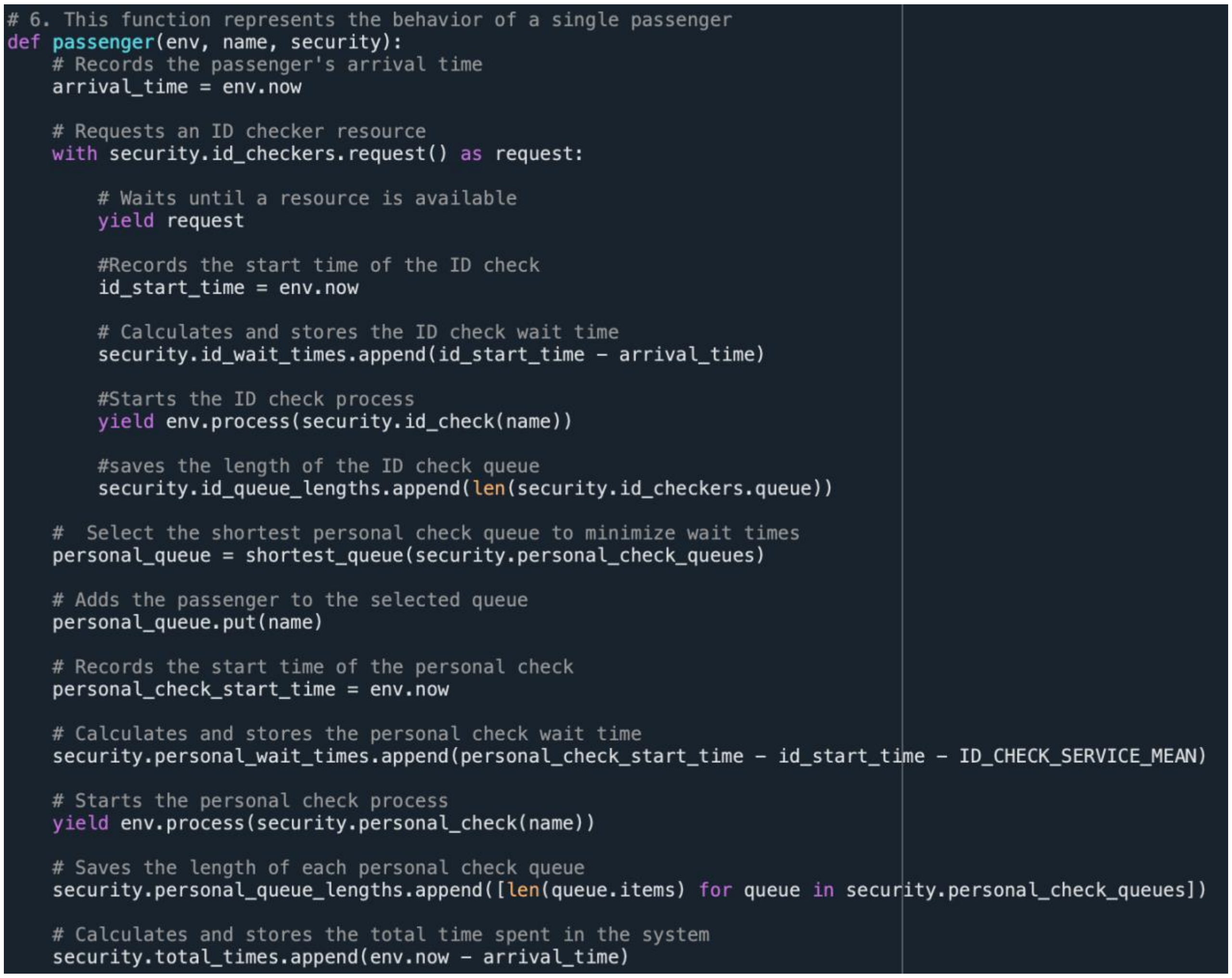


```python
# 5. This function simulates passenger arrivals
def passenger_arrival(env, security):
    passenger_count = 0

    # Creates an infinite loop to simulate continuous arrivals
    while True:

        # Generates the time until the next arrival using an exponential distribution
        yield env.timeout(random.expovariate(ARRIVAL_RATE))

        # Increase the passenger count
        passenger_count += 1

        # Create a new passenger process
        env.process(passenger(env, f"Passenger {passenger_count}", security))
```

Step 6: Create a function to represent the behavior of a single passenger:

Each passenger goes through the following steps:

1. Arrives at the ID check queue, and waits for an available checker.
2. Records wait time before being checked.
3. Undergoes ID check, which takes a random amount of time.
4. Chooses the shortest personal check queue and enters it.
5. Waits for personal check and records that wait time.
6. Goes through the personal check, which takes 0.5–1 minute.
7. Records total time spent in the system before leaving.

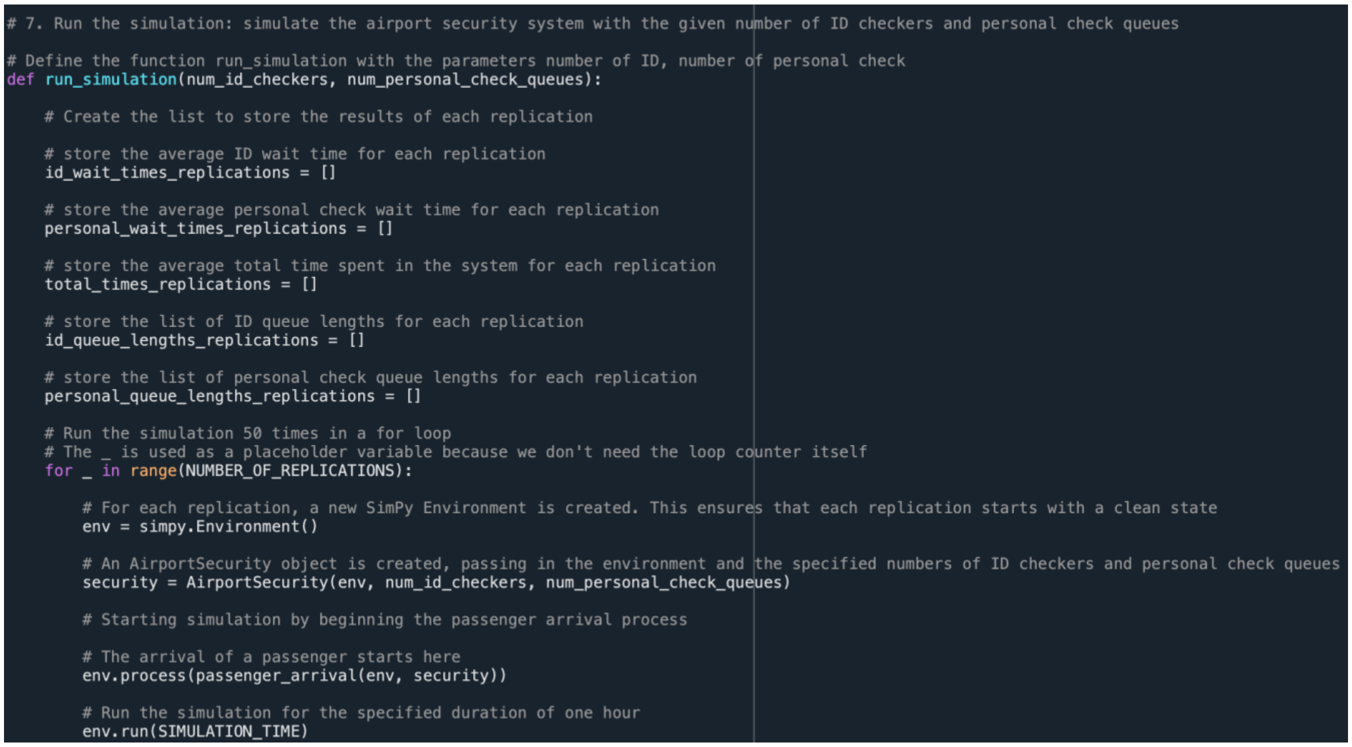


```python
# 6. This function represents the behavior of a single passenger
def passenger(env, name, security):
    # Records the passenger's arrival time
    arrival_time = env.now

    # Requests an ID checker resource
    with security.id_checkers.request() as request:

        # Waits until a resource is available
        yield request

        #Records the start time of the ID check
        id_start_time = env.now

        # Calculates and stores the ID check wait time
        security.id_wait_times.append(id_start_time - arrival_time)

        #Starts the ID check process
        yield env.process(security.id_check(name))

        #saves the length of the ID check queue
        security.id_queue_lengths.append(len(security.id_checkers.queue))

    #  Select the shortest personal check queue to minimize wait times
    personal_queue = shortest_queue(security.personal_check_queues)

    # Adds the passenger to the selected queue
    personal_queue.put(name)

    # Records the start time of the personal check
    personal_check_start_time = env.now

    # Calculates and stores the personal check wait time
    security.personal_wait_times.append(personal_check_start_time - id_start_time - ID_CHECK_SERVICE_MEAN)

    # Starts the personal check process
    yield env.process(security.personal_check(name))

    # Saves the length of each personal check queue
    security.personal_queue_lengths.append([len(queue.items) for queue in security.personal_check_queues])

    # Calculates and stores the total time spent in the system
    security.total_times.append(env.now - arrival_time)
```

Step 7: Run the simulation: simulate the airport security system with the given number of ID checkers and personal check queues:

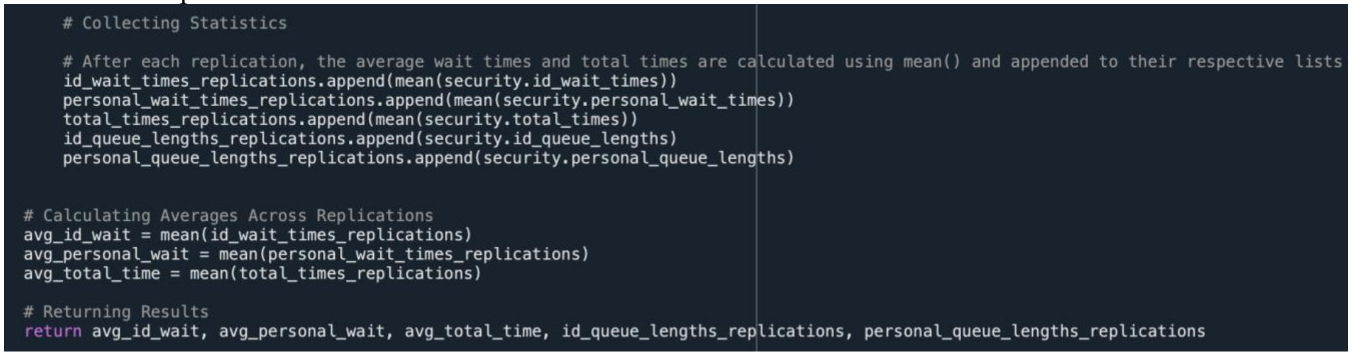
This function controls the entire simulation:

1. Runs the simulation 50 times for accuracy.
2. Creates a new environment (simpy.Environment()) for each run.
3. Initializes a new AirportSecurity system with the given number of ID checkers and personal queues.
4. Starts the passenger arrival process in the simulation.
5. Runs the simulation for 1 hour (env.run(SIMULATION_TIME)).
6. Collects average wait times and queue lengths for analysis.



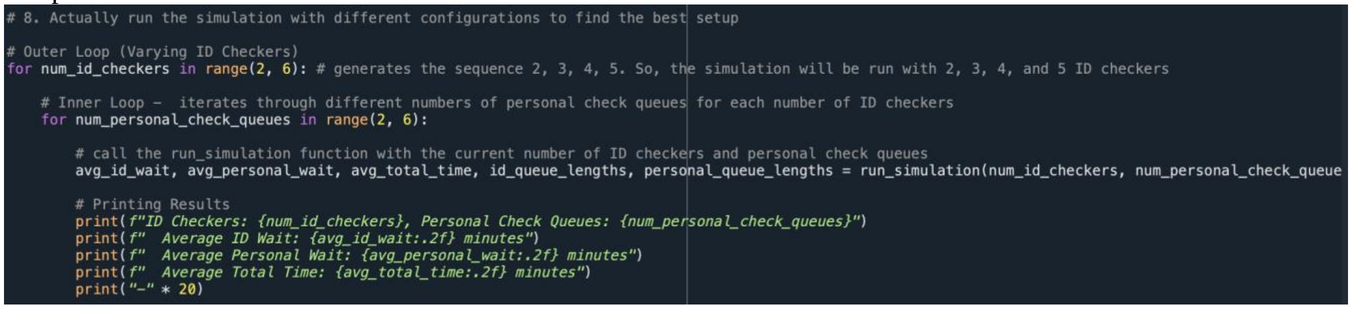
```python
# 7. Run the simulation: simulate the airport security system with the given number of ID checkers and personal check queues

# Define the function run_simulation with the parameters number of ID, number of personal check
def run_simulation(num_id_checkers, num_personal_check_queues):

    # Create the list to store the results of each replication

    # store the average ID wait time for each replication
    id_wait_times_replications = []

    # store the average personal check wait time for each replication
    personal_wait_times_replications = []

    # store the average total time spent in the system for each replication
    total_times_replications = []

    # store the list of ID queue lengths for each replication
    id_queue_lengths_replications = []

    # store the list of personal check queue lengths for each replication
    personal_queue_lengths_replications = []

    # Run the simulation 50 times in a for loop
    # The _ is used as a placeholder variable because we don't need the loop counter itself
    for _ in range(NUMBER_OF_REPLICATIONS):

        # For each replication, a new SimPy Environment is created. This ensures that each replication starts with a clean state
        env = simpy.Environment()

        # An AirportSecurity object is created, passing in the environment and the specified numbers of ID checkers and personal check queues
        security = AirportSecurity(env, num_id_checkers, num_personal_check_queues)

        # Starting simulation by beginning the passenger arrival process

        # The arrival of a passenger starts here
        env.process(passenger_arrival(env, security))

        # Run the simulation for the specified duration of one hour
        env.run(SIMULATION_TIME)
```

…continue Step 7…



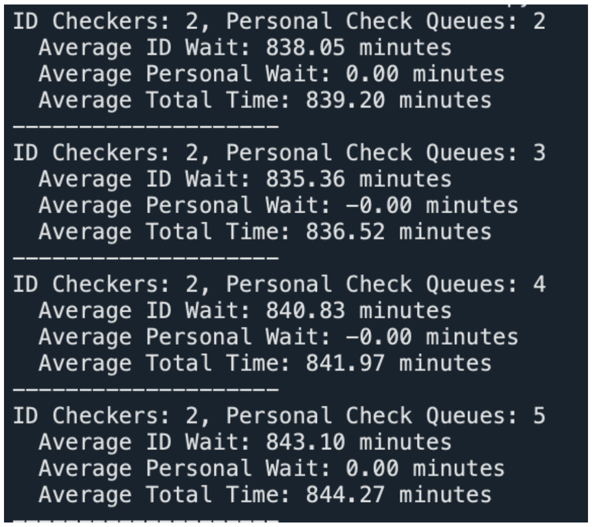
```python
        # Collecting Statistics

        # After each replication, the average wait times and total times are calculated using mean() and appended to their respective lists
        id_wait_times_replications.append(mean(security.id_wait_times))
        personal_wait_times_replications.append(mean(security.personal_wait_times))
        total_times_replications.append(mean(security.total_times))
        id_queue_lengths_replications.append(security.id_queue_lengths)
        personal_queue_lengths_replications.append(security.personal_queue_lengths)


    # Calculating Averages Across Replications
    avg_id_wait = mean(id_wait_times_replications)
    avg_personal_wait = mean(personal_wait_times_replications)
    avg_total_time = mean(total_times_replications)

    # Returning Results
    return avg_id_wait, avg_personal_wait, avg_total_time, id_queue_lengths_replications, personal_queue_lengths_replications
```

Step 8:
At the end, you would call run_simulation(num_id_checkers, num_personal_check_queues) to test different setups.

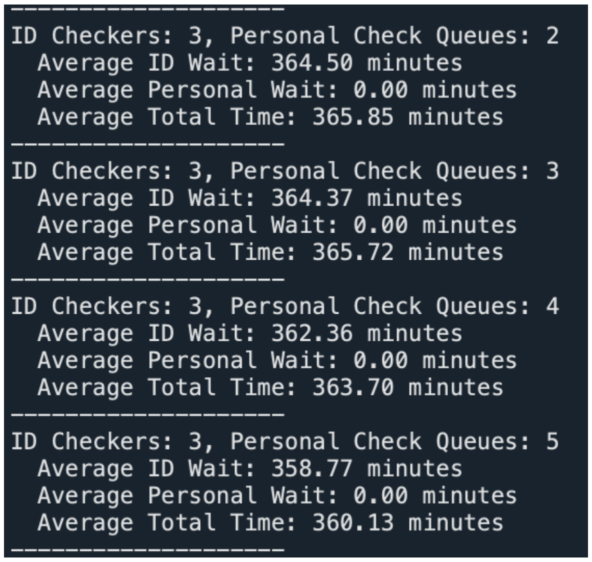


```
# 8. Actually run the simulation with different configurations to find the best setup

# Outer Loop (Varying ID Checkers)
for num_id_checkers in range(2, 6): # generates the sequence 2, 3, 4, 5. So, the simulation will be run with 2, 3, 4, and 5 ID checkers

    # Inner Loop -  iterates through different numbers of personal check queues for each number of ID checkers
    for num_personal_check_queues in range(2, 6):

        # call the run_simulation function with the current number of ID checkers and personal check queues
        avg_id_wait, avg_personal_wait, avg_total_time, id_queue_lengths, personal_queue_lengths = run_simulation(num_id_checkers, num_personal_check_queue

        # Printing Results
        print(f"ID Checkers: {num_id_checkers}, Personal Check Queues: {num_personal_check_queues}")
        print(f"  Average ID Wait: {avg_id_wait:.2f} minutes")
        print(f"  Average Personal Wait: {avg_personal_wait:.2f} minutes")
        print(f"  Average Total Time: {avg_total_time:.2f} minutes")
        print("-" * 20)
```

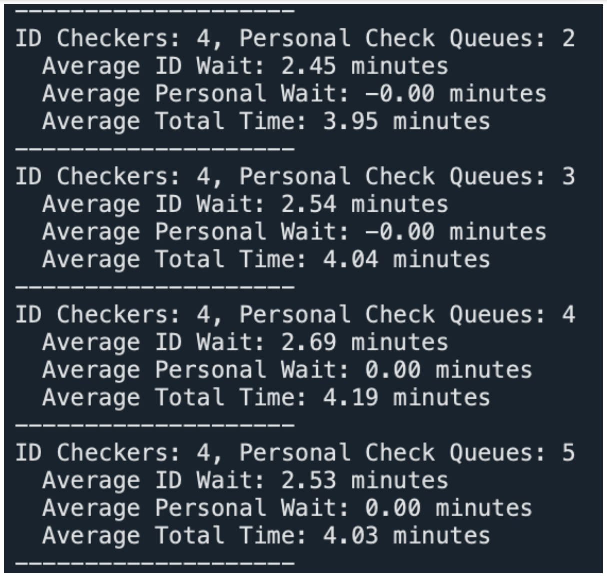
Results:



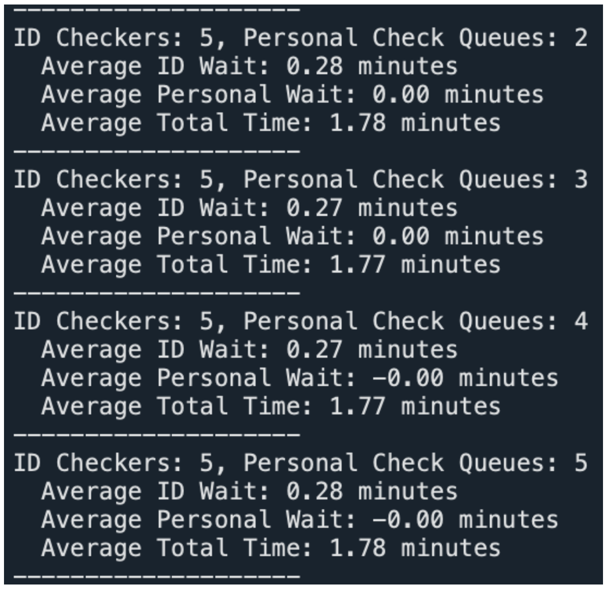
```
ID Checkers: 2, Personal Check Queues: 2
  Average ID Wait: 838.05 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 839.20 minutes
--------------------
ID Checkers: 2, Personal Check Queues: 3
  Average ID Wait: 835.36 minutes
  Average Personal Wait: -0.00 minutes
  Average Total Time: 836.52 minutes
--------------------
ID Checkers: 2, Personal Check Queues: 4
  Average ID Wait: 840.83 minutes
  Average Personal Wait: -0.00 minutes
  Average Total Time: 841.97 minutes
--------------------
ID Checkers: 2, Personal Check Queues: 5
  Average ID Wait: 843.10 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 844.27 minutes
--------------------
```



```
--------------------
ID Checkers: 3, Personal Check Queues: 2
  Average ID Wait: 364.50 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 365.85 minutes
--------------------
ID Checkers: 3, Personal Check Queues: 3
  Average ID Wait: 364.37 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 365.72 minutes
--------------------
ID Checkers: 3, Personal Check Queues: 4
  Average ID Wait: 362.36 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 363.70 minutes
--------------------
ID Checkers: 3, Personal Check Queues: 5
  Average ID Wait: 358.77 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 360.13 minutes
--------------------
```



```
--------------------
ID Checkers: 4, Personal Check Queues: 2
  Average ID Wait: 2.45 minutes
  Average Personal Wait: -0.00 minutes
  Average Total Time: 3.95 minutes
--------------------
ID Checkers: 4, Personal Check Queues: 3
  Average ID Wait: 2.54 minutes
  Average Personal Wait: -0.00 minutes
  Average Total Time: 4.04 minutes
--------------------
ID Checkers: 4, Personal Check Queues: 4
  Average ID Wait: 2.69 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 4.19 minutes
--------------------
ID Checkers: 4, Personal Check Queues: 5
  Average ID Wait: 2.53 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 4.03 minutes
--------------------
```



```
--------------------
ID Checkers: 5, Personal Check Queues: 2
  Average ID Wait: 0.28 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 1.78 minutes
--------------------
ID Checkers: 5, Personal Check Queues: 3
  Average ID Wait: 0.27 minutes
  Average Personal Wait: 0.00 minutes
  Average Total Time: 1.77 minutes
--------------------
ID Checkers: 5, Personal Check Queues: 4
  Average ID Wait: 0.27 minutes
  Average Personal Wait: -0.00 minutes
  Average Total Time: 1.77 minutes
--------------------
ID Checkers: 5, Personal Check Queues: 5
  Average ID Wait: 0.28 minutes
  Average Personal Wait: -0.00 minutes
  Average Total Time: 1.78 minutes
--------------------
```

Interpret the result:

Look at the result, we can see that the number of ID checkers is the key to the wait time.

If the ID Checkers are less than 4, no matter how the Personal Check Queues change from 2 to 5, the average wait time is greater than 15 minutes.

Therefore, to make sure the average wait times is less than 15 minutes, the ID Checkers must be greater than 3.