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 λ1 = 5 per minute (i.e., mean interarrival rate ⍰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 ⍰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 λ1 = 50 to simulate a busier airport.]

Used SimPy model for this system.

Built the Model first and used the model to determine how many scanners and how many boarding-pass checkers are required. Using various numbers in the simulation, run each, and see what the results are.

 Here's the Final average wait time (across 100 replications) for each combination of boarding-pass checkers and scanners between 35 and 40:

```
----- 37 -- 36 -----

Average system time = 16.78

Average check time = 0.75

Average scan time = 0.75

Average wait time = 15.28

----- 37 -- 37 -----

Average system time = 9.53

Average check time = 0.75

Average scan time = 0.75

Average wait time = 8.04
```

----- 37 -- 38 -----

Average system time = 8.15

Average check time = 0.75

Average scan time = 0.75

Average wait time = 6.65

Based on the runs, it looks like 37 boarding-pass checkers and 37 scanners is sufficient to get average waiting time below 15 minutes (all the way down to 8 minutes). Refer the code below

```
####### Question 13.2 - using Simpy ######

# ---------- Import modules -----------

#import SimPy module

import simpy


# Import random module

import random

# ------------ Set constants ---------------


numCheckers = 35 # smallest number of boarding-pass checkers

numScanners = 35 # smallest number of scanners


arrRate = 50 # arrival rate (passengers per minute)

checkRate = 0.75 # boarding-pass check rate (minutes per passenger)

minScan = 0.5 # scanner minimum time for uniform distribution

maxScan = 1.0 # scanner maximum time for uniform distribution

runTime = 720 # run time (minutes) per simulation

replications = 100 # number of replications
```

```
# ------------ Initialize global variables ----------

avgCheckTime = [[0.0 for i in range(6)] for j in range(6)] # average boarding-pass check time (for each replication)

avgScanTime = [[0.0 for i in range(6)] for j in range(6)] # average scan time (for each replication)

avgWaitTime = [[0.0 for i in range(6)] for j in range(6)] # average total wait time (for each replication)

avgSystemTime = [[0.0 for i in range(6)] for j in range(6)] # average total time in system (for each replication)


# ------------ Create model ------------------

# System class


class System(object):

  def __init__(self,env,i,j):

    self.env = env

    self.checker = simpy.Resource(env,i+numCheckers) # define number of boarding-pass checkers

    self.scanner = [] # define a set of scanners with 1 each; needed because each has its own queue

    for i in range(j+numScanners):

      self.scanner.append(simpy.Resource(env,1))


  # define boarding-pass check time (exponential)

  def check(self,passenger):

    # For some reason in python, expovariate actually uses 1 over the mean, like Poisson

    yield self.env.timeout(random.expovariate(1.0/checkRate))


  # define scan time (uniform)

  def scan(self,passenger):

    yield self.env.timeout(random.uniform(minScan,maxScan))


# Passenger process through system


def passenger(env,name,s,i,j,pnum):
```

```
# access global variables to be able to modify them
global checkWait
global scanWait
global sysTime
global totThrough

timeArrive = env.now # note arrival time of passenger



# print('%s arrives at time %.2f' % (name,timeArrive))

# Go through boarding-pass check queue
with s.checker.request() as request:
   # print('check queue length = %d' % len(s.checker.queue))
   yield request # request a checker
   tIn = env.now # note when passenger starts being checked
   yield env.process(s.check(name)) # call check process
   tOut = env.now # note when passenger ends being checked
   checkTime[pnum] = (tOut - tIn) # calculate total time for passenger to be checked

# Find the shortest scanner queue (note: scanners are numbered 0 through numScanners-1)
minq = 0
for k in range(1,j+numScanners):
   if (len(s.scanner[k].queue) < len(s.scanner[minq].queue)):
      minq = k

# print('scanner queue %d lengths = %d' % (minq,len(s.scanner[minq].queue)))

# Go through scanner queue
with s.scanner[minq].request() as request: # use scanner number minq (the shortest, from above)
```

```
        yield request # request the scanner

        tIn = env.now # note when passenger starts being scanned

        yield env.process(s.scan(name)) # call scan process

        tOut = env.now # note when passenger ends being scanned

        scanTime[pnum] = (tOut - tIn) # calculate total time for passenger to be scanned


    timeLeave = env.now # note time passenger finishes

    sysTime[pnum] = (timeLeave - timeArrive) # calculate total time in system for passenger

    totThrough += 1 # count another passenger who got through the system




# Passenger arrival process


def setup(env,i,j):

    k = 0

    s = System(env,i,j)

    while True: # keep doing it (until simulation ends)

        yield env.timeout(random.expovariate(arrRate)) # find tieme until next passenger is created

        k += 1 # count one more passenger


        # send the passenger through its process

        env.process(passenger(env,'Passenger %d' % k,s,i,j,k)) # name the passenger "Passenger i"




# ------------------ Run the model --------------------


for i in range(6): # number of boarding-pass checkers

    for j in range(6): # number of scanners


        # for each replication
    for k in range(replications):

        # choose random seed
        random.seed(k)

        # create environment
        env = simpy.Environment()

        # initialize global variables
        totThrough = 0
        checkTime = [0.0] * int(arrRate*runTime*1.5)
        scanTime = [0.0] * int(arrRate*runTime*1.5)
        sysTime = [0.0] * int(arrRate*runTime*1.5)

        # run the simulation
        env.process(setup(env,i,j)) # start passenger arrival process
        env.run(until=runTime) # run for runTime simulated minutes

        # Calculate average times for this replication

        # print('%d : Replication %d times %.2f %.2f %.2f' % (totThrough,k+1,sum(sysTime[1:totThrough]) /
totThrough,sum(checkTime[1:totThrough]) / totThrough,sum(scanTime[1:totThrough]) / totThrough))

        avgSystemTime[i][j] += (sum(sysTime[1:totThrough]) / totThrough)
        avgCheckTime[i][j] += (sum(checkTime[1:totThrough]) / totThrough)
        avgScanTime[i][j] += (sum(scanTime[1:totThrough]) / totThrough)

    avgWaitTime[i][j] = (avgSystemTime[i][j] - avgCheckTime[i][j] - avgScanTime[i][j])


    # Calculate overall averages across all replications
```

```
avgSystemTime[i][j] /= replications

avgCheckTime[i][j] /= replications

avgScanTime[i][j] /= replications

avgWaitTime[i][j] /= replications


print('----- %d -- %d -----' % (i+35,j+35))

print('Average system time = %.2f' % avgSystemTime[i][j])

print('Average check time = %.2f' % avgCheckTime[i][j])

print('Average scan time = %.2f' % avgScanTime[i][j])

print('Average wait time = %.2f' % avgWaitTime[i][j])
```