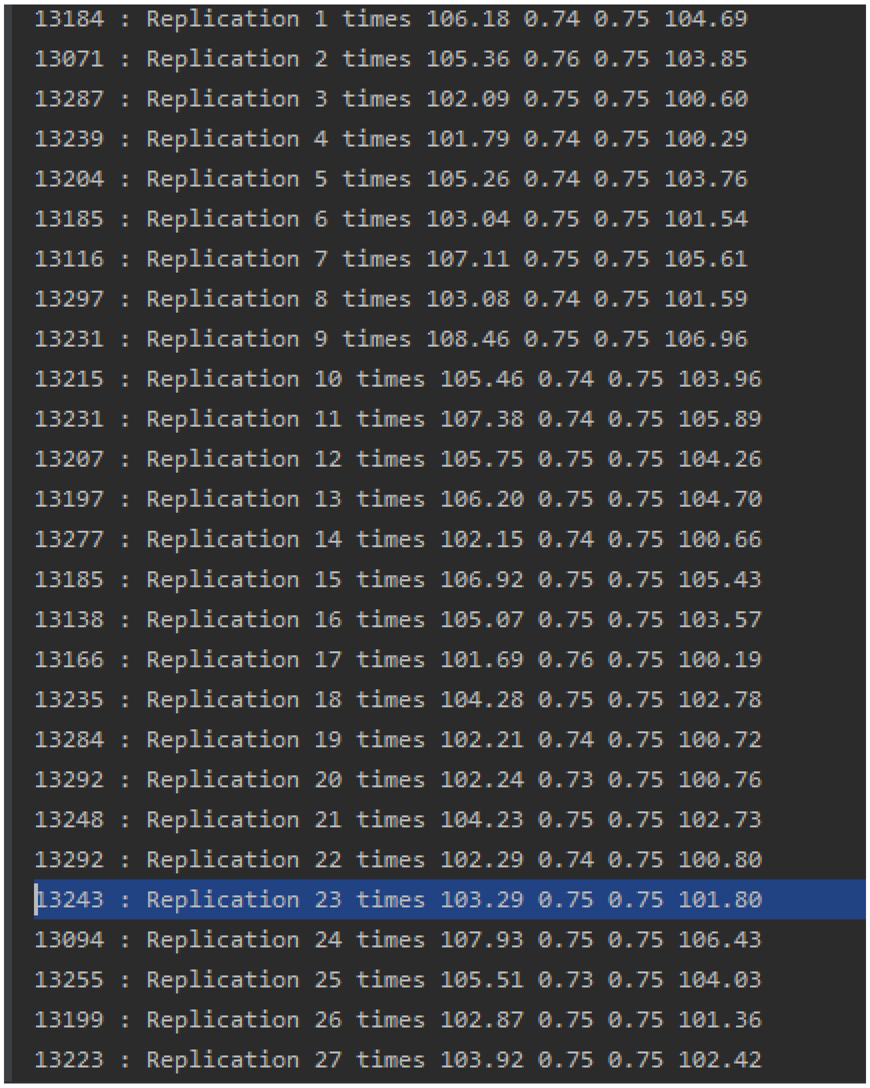
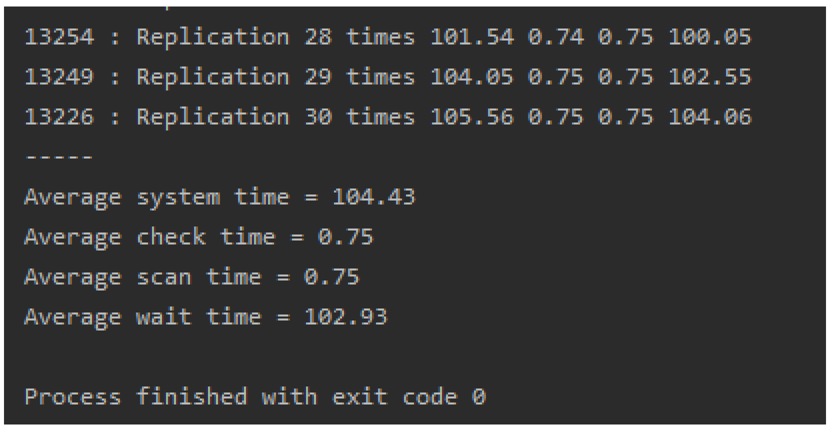
```
#
#Question 13.2

#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.]

# Import Libray

# import SimPy module
import simpy

# Import random module
import random # probability distributions

# Set constants based on requirement

boardcheck = 20  # number of boarding-pass checkers
Scanners = 20  # number of scanners

arrival = 45  # 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 = 500  # run time (minutes) per simulation
replications = 30  # number of replications

#Initialize  variables

avgCheckTime = []  # average boarding-pass check time (for each replication)
avgScanTime = []  # average scan time (for each replication)
avgWaitTime = []  # average total wait time (for each replication)
avgSystemTime = []  # average total time in system (for each replication)


# Create model
# Setup Entities
```

```python
class Entity(object):
    def __init__(self, env):
        self.env = env
        self.checker = simpy.Resource(env, boardcheck)  # 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(Scanners):
            self.scanner.append(simpy.Resource(env, 1))

    # define boarding-pass check time (exponential)
    def check(self, passenger):
        # expovariate actually uses 1 over the mean, like Poisson
        yield self.env.timeout(random.expovariate(1.0 / checkRate))# using yield as
simpy is a generator

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

    timeArrive = env.now  # note arrival time of passenger(in sec)

    # 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.append(tOut - tIn)  # calculate total time for passenger to be
checked

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

    # 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.append(tOut - tIn)  # calculate total time for passenger to be
scanned

    timeLeave = env.now  # note time passenger finishes
    sysTime.append(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 = 0
    s = Entity(env)
    while True:  # keep doing it (until simulation ends)
        yield env.timeout(random.expovariate(arrival))  # find time until next
passenger is created
        i += 1  # count one more passenger

        # send the passenger through its process
        env.process(passenger(env, 'Passenger %d' % i, s))  # name the passenger
"Passenger i"


# steps to run

# for each replication
for i in range(replications):
    # choose random seed
    random.seed(i)

    # create environment (all processes are run within the environment
    env = simpy.Environment()

    # initialize global variables
    totThrough = 0
    checkTime = []
    scanTime = []
    sysTime = []

    # run the simulation
    env.process(setup(env))  # start passenger arrival process
    env.run(until=runTime)  # run for runTime simulated minutes (720)

    # Calculate average times for this replication

    avgSystemTime.append(sum(sysTime[1:totThrough]) / totThrough)
    avgCheckTime.append(sum(checkTime[1:totThrough]) / totThrough)
    avgScanTime.append(sum(scanTime[1:totThrough]) / totThrough)
    avgWaitTime.append(avgSystemTime[i] - avgCheckTime[i] - avgScanTime[i])
    print('%d : Replication %d times %.2f %.2f %.2f %.2f' % (
    totThrough, i + 1, avgSystemTime[i], avgCheckTime[i], avgScanTime[i],
avgWaitTime[i]))

# Calculate overall averages across all replications

print('-----')
print('Average system time = %.2f' % (sum(avgSystemTime) / replications))
print('Average check time = %.2f' % (sum(avgCheckTime) / replications))
print('Average scan time = %.2f' % (sum(avgScanTime) / replications))
print('Average wait time = %.2f' % (sum(avgWaitTime) / replications))
```

Results of the simulation process:



```
13184 : Replication 1 times 106.18 0.74 0.75 104.69
13071 : Replication 2 times 105.36 0.76 0.75 103.85
13287 : Replication 3 times 102.09 0.75 0.75 100.60
13239 : Replication 4 times 101.79 0.74 0.75 100.29
13204 : Replication 5 times 105.26 0.74 0.75 103.76
13185 : Replication 6 times 103.04 0.75 0.75 101.54
13116 : Replication 7 times 107.11 0.75 0.75 105.61
13297 : Replication 8 times 103.08 0.74 0.75 101.59
13231 : Replication 9 times 108.46 0.75 0.75 106.96
13215 : Replication 10 times 105.46 0.74 0.75 103.96
13231 : Replication 11 times 107.38 0.74 0.75 105.89
13207 : Replication 12 times 105.75 0.75 0.75 104.26
13197 : Replication 13 times 106.20 0.75 0.75 104.70
13277 : Replication 14 times 102.15 0.74 0.75 100.66
13185 : Replication 15 times 106.92 0.75 0.75 105.43
13138 : Replication 16 times 105.07 0.75 0.75 103.57
13166 : Replication 17 times 101.69 0.76 0.75 100.19
13235 : Replication 18 times 104.28 0.75 0.75 102.78
13284 : Replication 19 times 102.21 0.74 0.75 100.72
13292 : Replication 20 times 102.24 0.73 0.75 100.76
13248 : Replication 21 times 104.23 0.75 0.75 102.73
13292 : Replication 22 times 102.29 0.74 0.75 100.80
13243 : Replication 23 times 103.29 0.75 0.75 101.80
13094 : Replication 24 times 107.93 0.75 0.75 106.43
13255 : Replication 25 times 105.51 0.73 0.75 104.03
13199 : Replication 26 times 102.87 0.75 0.75 101.36
13223 : Replication 27 times 103.92 0.75 0.75 102.42
```



```
13254 : Replication 28 times 101.54 0.74 0.75 100.05
13249 : Replication 29 times 104.05 0.75 0.75 102.55
13226 : Replication 30 times 105.56 0.75 0.75 104.06
-----
Average system time = 104.43
Average check time = 0.75
Average scan time = 0.75
Average wait time = 102.93

Process finished with exit code 0
```