**CSC148 – Quiz#4 – Tuesday Nov 26, 2019, Name** \_\_\_Answers\_\_\_ Score max: 17\_\_\_\_\_

Closed notes; NO mobile devices/calculators can be in-use during quiz

UNDERLINE all answers for questions that do not provide an answer area “\_\_\_\_\_\_\_” Graders will not search for answers

**Q#1. 5 pts** Answer either True/False, or Fill-in-the-blanks, as directed by each question part

**1 pt.** Answer True/False: \_\_True\_\_ A simpy process function must contain at least one yield statement

**2 pts** Fill-in-the-blanks: In the underlined section that follows, code a simpy expression whose numeric value is

3 t.u.s more than (current model time) \_\_env.now + 3\_\_

**1 pt** Answer True/False: \_False\_\_Assuming that module simpy has been imported, the following statement is illegal in a

simpy simulation: myenv = simpy.Environment()

**1 pt** Answer True/False: \_False\_ In a simpy simulation script, it is illegal to call a Python function that is not a process

function

**Q#2** **5 pts** Write correct simpy statements that assigns variable names to each of 5 instances of a process function named xyz that is being scheduled. xyz has no parameters. Each of the 5 instance variable names is an element of a Python list named pfList. Prior to the instance creations, pfList does not exist.

pfList = [] # Initialize references list

for k in range(5):

pfList.append(env.process(xyz()))

Here is a complete simulation illustrating the above pfList processing

""" Create a list of pf scheduling calls demo\_schedCalls.py """

import pdb

import simpy # Import simpy environment

env = simpy.Environment()

def xyz(env):

“”” Dummy pf illustrating that has several instances scheduled “””

print("Instance of pf xyz starting at time ",env.now)

yield env.timeout(0) # Do nothing

pfList = [] # Init list of pf schediling calls

def sched\_pfs(env):

for k in range(5):

yield env.timeout(1)

pfList.append(env.process(xyz(env)))

print("Next scheduled xyz instance ",pfList[k]," started at time ",env.now)

env.process(sched\_pfs(env))

env.run()

print("Simmulation finished at time ",env.now)

=========== RESTART: C:/Users/bill/148\_f19/quiz/demo\_schedCalls.py ===========

Next scheduled xyz instance <Process(xyz) object at 0x1fc86179438> started at time 1

Instance of pf xyz starting at time 1

Next scheduled xyz instance <Process(xyz) object at 0x1fc861790f0> started at time 2

Instance of pf xyz starting at time 2

Next scheduled xyz instance <Process(xyz) object at 0x1fc861790b8> started at time 3

Instance of pf xyz starting at time 3

Next scheduled xyz instance <Process(xyz) object at 0x1fc8606db70> started at time 4

Instance of pf xyz starting at time 4

Next scheduled xyz instance <Process(xyz) object at 0x1fc860f4ef0> started at time 5

Instance of pf xyz starting at time 5

Simmulation finished at time 5

>>>

Note – if fcn xyz does NOT contain at least one yield statement, then xyz is NOT A pf, as illustrated by this error msg::

Traceback (most recent call last):

File "C:\Users\bill\148\_f19\quiz\demo\_schedCalls.py", line 15, in sched\_pfs

pfList.append(env.process(xyz(env)))

File "C:\Users\bill\AppData\Local\Programs\Python\Python36\lib\site-packages\simpy\events.py", line 311, in \_\_init\_\_

raise ValueError('%s is not a generator.' % generator)

ValueError: None is not a generator.

**Q#3 2 pts** A simpy simulation is required to execute for 250 model t.u.s. The events that are scheduled at time 250 must not be executed. Code the necessary simpy statement(s) to specify this model run termination requirement.

You do not need to code a complete model; code only what is necessary to specify when the simulation terminates.

env.run(until=250) Termination is controlled by env.run() and the parameters specified for the run() call.

**Q#4. 5 pts** Code a complete & correct simpy simulation script for the following specification.

All imports and related simpy initializations must be coded.

A process function (pf) models an assembly-line task “T” that must be repeated 5 times. Each T repetition takes 10 t.u. to finish. Each model time when the jth T repetition finishes, pf execution displays a message, depending on j like this:

Finished task T at time 20 *< -- The message displayed when the 2nd T repetition finishes*

Schedule the pf to execute exactly one time at model time 0. You do not need to name the pf instance.

After the pf instance finishes execution, print the message:

T executions finished at time xxx, where “xxx” is the model time when the pf instance finishes execution.

Also, specify that the model run terminates when there are no more events scheduled.

**Source code solution:**

""" q4q4f19.py """

import simpy

import pdb

env = simpy.Environment()

def doTask():

for k in range (5):

yield env.timeout(10)

print("Finished task T at time ", env.now)

env.process(doTask())

env.run()

print("T executions finished at time ",env.now)

===========

=============== RESTART: C:\Users\bill\148\_f19\quiz\q4q4f19.py =============== **< Execution demo**

Finished task T at time 10

Finished task T at time 20

Finished task T at time 30

Finished task T at time 40

Finished task T at time 50

T executions finished at time 50

>>>

**Some examples of illegal Python/simpy source code seen in quiz#4 papers:**

1 ) >>> p++

SyntaxError: invalid syntax As Intro Python notes file says: there is no auto incr/decr syntax in Python

2 ) Variable scope issues:

var i = 1 < -- var is not a Python keyword; even if it was, and i had type int

3 ) Demo of some Python scope of variables properties (that are NOT THE SAME as in many other languages)

=================================================================

"""

Illegal variable use Demo demo\_illVarUse.py

Demo - Python’s default parameter passing mode is NOT like most other languages; it is “call by object reference”

"""

i = 2 # i and j are “global” variables

print("id(i) is ",id(i))

j = -4

def f():

global j

# print("Value of i is ",i) # < == Illegal reference to i before it is defined in f

# print("Value of j is ",j) # < == j references the global j because of the global statement

print("Start executing fcn f")

i = 3 # This i is a NEW LOCAL variable that is destroyed when f finishes

if i==3:

print("During f execution, i value, id, & type are ",i," ",id(i)," ",type(i))

j = 58 # Change value of global j

print("Finish executing fcn f")

def pf(env):

""" pf that repeats f()/yield/print code block 2 times """

print("Starting process fcn pf at time ",env.now)

for k in range(2):

f() # Calling a non-pf function in a pf

yield env.timeout(5)

print("Current model time is ",env.now)

f() # f called BEFORE process function pf calls f 2 times

print("Value of i after in-line call to f is ",i)

print(“Value of j after in-line call to f is ",j)

import simpy

env = simpy.Environment() # simpy environment refr

env.process(pf(env)) # Schedule an anonymous (i.e. un-named instance of pf

env.run()

print("Simulation finished at time ",env.now)

Execution results

=========== RESTART: C:\Users\bill\148\_f19\quiz\demo\_illvarUse.py ===========

id(i) is 1929884768

Start executing fcn f

During f execution, i value, id, & type are 3 1929884800 <class 'int'>

Finish executing fcn f

Value of i after in-line call to f is 2 < -- f execution did NOT change i’s value after return

Value of j after in-line call to f is 58 < -- f execution changed j’s value after return

Starting process fcn pf at time 0

Start executing fcn f

During f execution, i value, id, & type are 3 1929884800 <class 'int'>

Finish executing fcn f

Current model time is 5

Start executing fcn f

During f execution, i value, id, & type are 3 1929884800 <class 'int'>

Finish executing fcn f

Current model time is 10

Simulation finished at time 10

>>>