Timed Lab 1 – Supply Chain Analysis

Timed Lab Information:

This is a Timed Lab; this Timed Lab is worth 23 exam points.

For this Timed Lab, you *may* use

- Course notes
- Homeworks
- Recitation assignments
- Other course material
- Any material you may find on the Internet

However, you *may not*

- Communicate with other people via any means. This means no Facebook, cell phones, Google Talk, etc.

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistance with debugging code, program logic, etc.

You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends: No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

T-Square will not permit any late submissions; ensure that you submit your code to T-Square incrementally to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warning 10 and 5 minutes before the end of the recitation period; you should submit at these times.

In your collaboration statement, if you use code from somewhere that is *not* a class resource (i.e. not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the file.

Problem Description:

You and your programming intern work for a company with a very large supply chain. You have been asked by your pointy-haired manager to create a simple program that will allow employees to enter information about routes in the supply chain and perform some data analysis on those routes, specifically returning a list of the routes with a utilization less than some specified amount as well as being able to compute how much profit comes from each route (assuming each route only transports one product).

You have decided to accomplish this by writing (at least) four functions in Python (You may write additional helper functions if you desire):

1. getYesNo
2. getRoute
3. getRoutesList
4. filterRoutes

Fortunately, one of these five functions, getYesNo, has been written by the UGA programming intern that works for your company, but it would appear that the rest of these functions are beyond his expertise! His file that you should finish writing is located on your company Intranet (The Timed Lab 1 T-Square Assignment); download his file and add the remainder of the functions and submit your work when you are finished. Unfortunately, your boss needs this very fast; you only have 1 hour and 45 minutes to finish this program, so work efficiently. Below are the detailed method descriptions your boss has given you to guide your development. You do not need to write any code that will automatically run your functions.

getYesNo:

This function has already been written for you by your intern. It takes in one parameter, a message, displays this to the user as a prompt, at which time the user must enter either yes or no (or some variant of YES or nO). It will return True if the user enters yes (or some variant) or False if the user enters no (or some variant) and will continue prompting the user until a valid yes or no is entered. You should not modify this function.

getRoute:

This function, which takes no parameters, will ask the user to enter information about one particular route in the supply chain. Specifically, it will ask the user to enter three pieces of data:

1. The source/origin (a string)
2. The destination (a string)
3. The number of goods transported over this route (an integer)

You may assume that the user enters valid data for each of the above; you do not need to error check (i.e. the number of goods will be an integer greater than or equal to 0). Once these three pieces of data are entered, you should return a tuple where the first item in the tuple is the source, the second item in the tuple is the destination, and the last item in the tuple is the number of goods transported over the route. Note that the last item in the tuple must be an integer!

Test Case:

getRoute()
Enter the source: ABC
Enter the destination: XYZ
Enter the amount transported: 400
Returns (‘ABC’, ‘XYZ’, 400)

getRoutesList:

This function, which takes no parameters, will create a list of routes. Start with an empty list. You are required to use your getRoute function that you wrote previously to get a route tuple from the user. Place this tuple into your list of routes, and then use the getYesNo function to ask the user if he/she wishes to enter another route. If the user wishes to enter another route, repeat the above process. If the user does not wish to enter another route (enters "no"), return the list of route tuples that you have collected so far.

Test Case:

getRoutesList()
Enter the source: ABC
Enter the destination: XYZ
Enter the amount transported: 400
Enter another route? Yes
Enter the source: DEF
Enter the destination: XYZ
Enter the amount transported: 1000
Enter another route? Yes
Enter the source: GHI
Enter the destination: ABC
Enter the amount transported: 450
Enter another route? No
Returns [(‘ABC’, ‘XYZ’, 400), (‘DEF’, ‘XYZ’, 1000), (‘GHI’, ‘ABC’, 450)]

filterRoutes:

The filterRoutes function takes in two parameters: the list of routes you want to filter and an integer which represents the amount transported cutoff or filter value. You should return a new list that contains all of the route tuples where the number of goods transported is **less than** the cutoff parameter. You should not modify the original list of route tuples.

Test Cases:

routesList = [(‘ABC’, ‘XYZ’, 400), (‘DEF’, ‘XYZ’, 1000), (‘GHI’, ‘ABC’, 450)]
x = filterRoutes(routesList, 500)
print(x)
[(‘ABC’, ‘XYZ’, 400), (‘GHI’, ‘ABC’, 450)] is printed

routesList = [(‘ABC’, ‘XYZ’, 400), (‘DEF’, ‘XYZ’, 1000), (‘GHI’, ‘ABC’, 450)]
x = filterRoutes(routesList, 200)
print(x)
[] is printed

routesList = [(‘ABC’, ‘XYZ’, 400), (‘DEF’, ‘XYZ’, 1000), (‘GHI’, ‘ABC’, 450)]
x = filterRoutes(routesList, 1050)
print(x)
[(‘ABC’, ‘XYZ’, 400), (‘DEF’, ‘XYZ’, 1000), (‘GHI’, ‘ABC’, 450)] is printed

print(routesList)
[(‘ABC’, ‘XYZ’, 400), (‘DEF’, ‘XYZ’, 1000), (‘GHI’, ‘ABC’, 450)] is printed. Note that the filterRoutes function did not modify the original routesList variable.

Grading:

getRoute	**6**
Function header correct	1
Returns a tuple	1
Correctly gets user input	2
Builds correct tuple	2
getRoutesList	**8**
Function header correct	1
Returns a list of tuples	1
Doesn’t duplicate getRoute/getYesNo	1
Correctly asks for/gets multiple routes	3
List of tuples returned is correct	2
filterRoutes	**9**
Function header correct	1
Returns a list of tuples	1
Properly checks all tuples for validity	3
Does not modify original list	1
New list contains correct tuples	3