Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A subsidiary organ of OIC

Laboratory Report: Lab 0

CSE 4712: Artificial Intelligence Lab

Name: Tasfia Tasneem Annesha
Student ID: 190041220
Section: 2B
Semester: 7th
Academic Year: 2021-2022

Date of Submission: 8/8/2023

1 Problem Statement

The task is divided into 3 questions, each of which calls for us to write a short piece of Python code to complete a specific task and then we need to check our tasks using an autograder file. The autograder gives each of the three questions a score of 3/3.

2 Analysis of The Problem

All three questions require the implementation of simple Python functions that test the learner's elementary implementational and problem-solving skills. A detailed analysis of each individual problem is given below:

2.1 Question 1: Addition

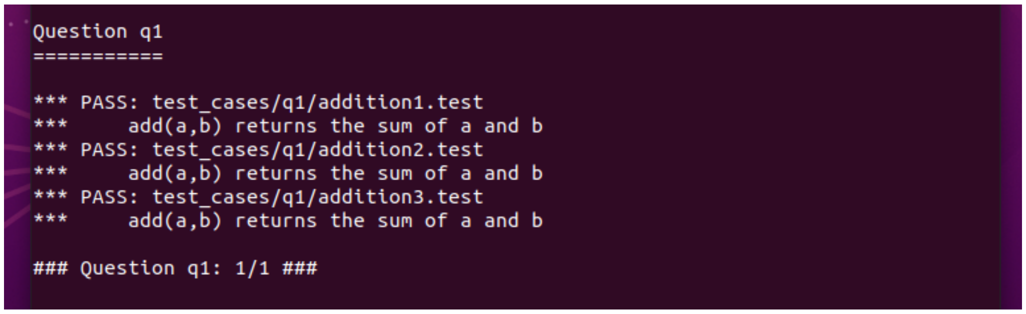
This task was very easy. In order to complete the task, we must sum the values of the two arguments passed to the function and return the result.

Working code:

```
def add(a, b):
   "Return the sum of a and b"
   "*** YOUR CODE HERE ***"
   return a+b
```

Explanation:

Returning a+b, where a and b are the arguments passed to the addition function, solves the problem in a straightforward manner. The time complexity is *O*(1).



```
Question q1
===========

*** PASS: test_cases/q1/addition1.test
***     add(a,b) returns the sum of a and b
*** PASS: test_cases/q1/addition2.test
***     add(a,b) returns the sum of a and b
*** PASS: test_cases/q1/addition3.test
***     add(a,b) returns the sum of a and b

### Question q1: 1/1 ###
```

2.2 Question 2: buyLotsOfFruit function

Working code:

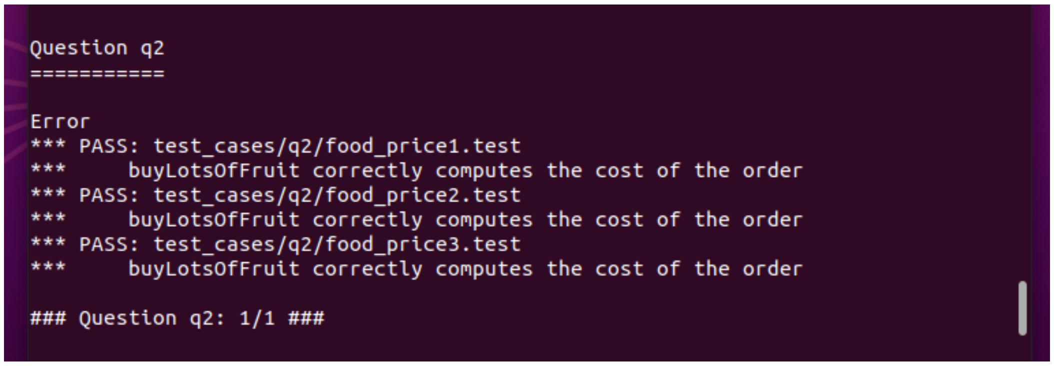
```
def buyLotsOfFruit(orderList):
    """
      orderList: List of (fruit, numPounds) tuples
Returns cost of order
    """
    totalCost = 0.0
    "*** YOUR CODE HERE ***"
    for fruit, numPounds in orderList:
        if fruit in fruitPrices:
            totalCost += numPounds * fruitPrices [fruit]
        else:
            print('Error')
            return None
    return totalCost
```

Analysis of the problem:

Here I had to figure out the overall price of all the fruits the customer purchased for this task. Here we are provided the function with an orderList list that contains the items and their corresponding quantities. Each item's price was listed in the fruitPrices dictionary.

Solution of the Problem:

The solution was as easy as iterating through each item in the **orderList** and using the **orderList** and **fruitPrices** to calculate the order's total cost. **Fruit** in the loop makes reference to the fruits listed in the **orderList**. **fruit** contains the fruit's name, which serves as the **fruitPrices** dictionary's key. The amount of the item in question is **numPounds**. **FruitPrices[fruit]** is used to find the price of a fruit, and its cost is calculated by multiplying it by the fruit's quantity. The **totalCost** value is returned after we calculate the price of each fruit and add them together.



```
Question q2
===========

Error
*** PASS: test_cases/q2/food_price1.test
***     buyLotsOfFruit correctly computes the cost of the order
*** PASS: test_cases/q2/food_price2.test
***     buyLotsOfFruit correctly computes the cost of the order
*** PASS: test_cases/q2/food_price3.test
***     buyLotsOfFruit correctly computes the cost of the order

### Question q2: 1/1 ###
```

2.3 Question 3: shopSmart function

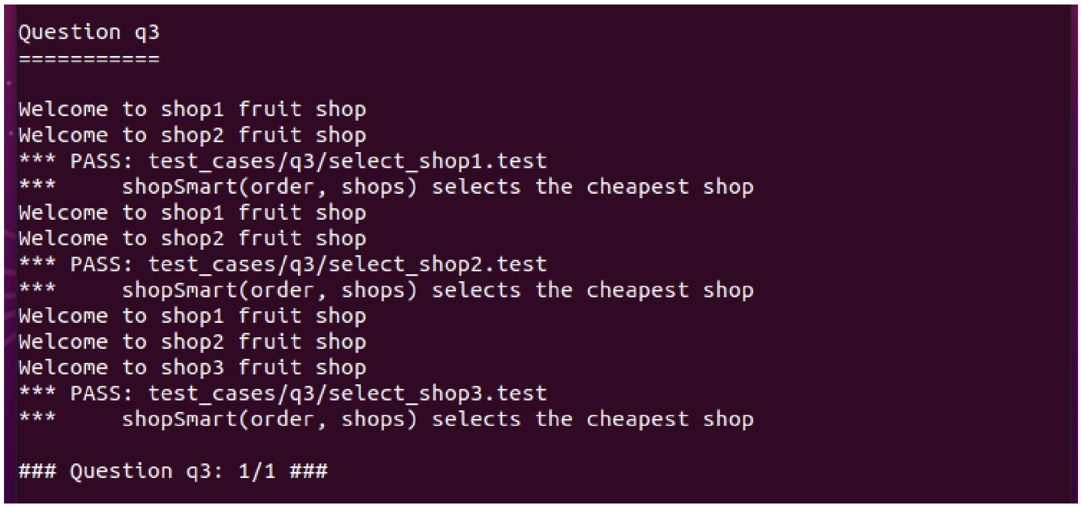
```
def shopSmart(orderList, fruitShops):
    """
        orderList: List of (fruit, numPound) tuples
        fruitShops: List of FruitShops
    """
    "*** YOUR CODE HERE ***"
    bestshop= None
    bestCost= float('inf')
    for shop in fruitShops:
        totalCost = 0.0
        totalCost += shop.getPriceOfOrder (orderList)
        if totalCost < bestCost:
            bestCost = totalCost
            bestshop=shop
    return bestshop
```

Explanation of the Question:

This question was very similar to the second. However, there was a further step. In this task, we had to choose the shop that would charge the least for our order from a list of shops.

Solution of the problem:

Here I first declared the **bestCost** variable infinity. Then using a for loop I have found the **totalCost** of each shop. If the **totalCost** is smaller than the **bestCost** then the bestCost is assigned totalCost. The **bestshop** value is returned after that.

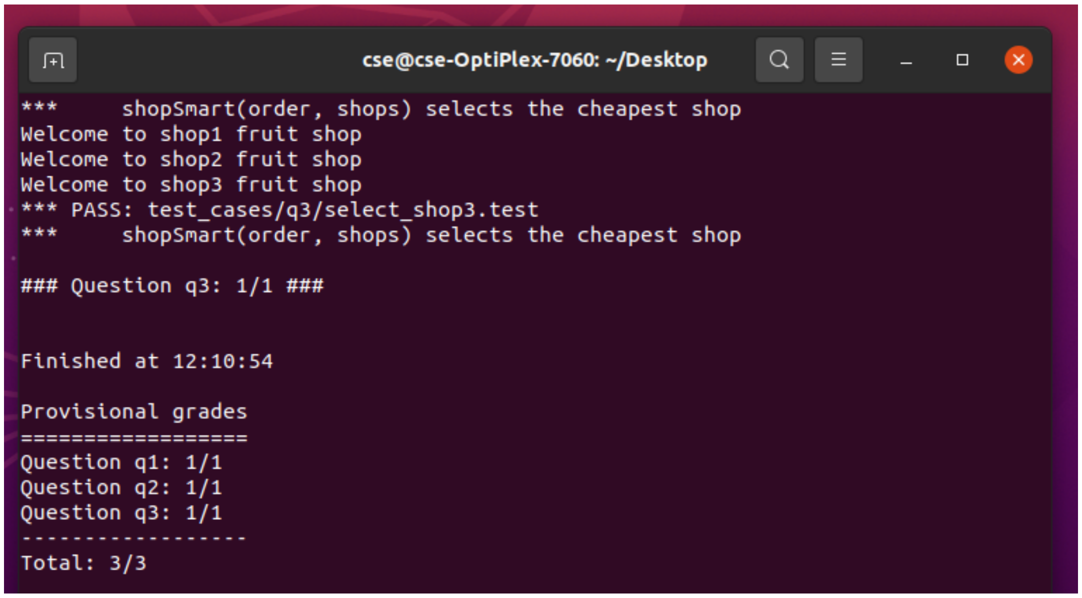


```
Question q3
===========

Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** PASS: test_cases/q3/select_shop1.test
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** PASS: test_cases/q3/select_shop2.test
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
Welcome to shop3 fruit shop
*** PASS: test_cases/q3/select_shop3.test
***     shopSmart(order, shops) selects the cheapest shop

### Question q3: 1/1 ###
```

Final Result:



```
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
Welcome to shop3 fruit shop
*** PASS: test_cases/q3/select_shop3.test
***     shopSmart(order, shops) selects the cheapest shop

### Question q3: 1/1 ###


Finished at 12:10:54

Provisional grades
==================
Question q1: 1/1
Question q2: 1/1
Question q3: 1/1
------------------
Total: 3/3
```