QUESTION 1 – ADDITION

This is solution code.

```python
def add(a, b):
    "Return the sum of a and b"
    "*** YOUR CODE HERE ***"
    print("Passed a=%s and b=%s, returning a+b=%s" % (a, b, a + b))
    return a + b
```

All I had to do was copy the code from the instruction pdf. But what actually I took as input from the code is I had to return a + b, so just did the a + b and returned it.

The problem with this type of declaration is, we can only pass two values to perform the addition.

QUESTION 2 – buyLotsOfFruit

This is solution code.

```python
def buyLotsOfFruit(orderList):
    """
        orderList: List of (fruit, numPounds) tuples

    Returns cost of order
    """
    totalCost = 0.0
    "*** YOUR CODE HERE ***"
    for order in orderList:
        fruit, pounds = order
        if fruit not in fruitPrices:
            print(f"Error: {fruit} is not in the fruitPrices list.")
            return None
        else:
            totalCost += fruitPrices[fruit] * pounds

    return totalCost
```

This function takes a list of tuples, where each tuple contains a fruit and a pound. It iterates through the list, and for each tuple, it checks if the fruit is in the fruitPrices list. If it is not, it prints an error message and returns None. If it is, it adds the cost of the fruit to the totalCost. Finally, it returns the totalCost.

This is just simple looping and condition checking.

QUESTION 3 – shopSmart

This is solution code.

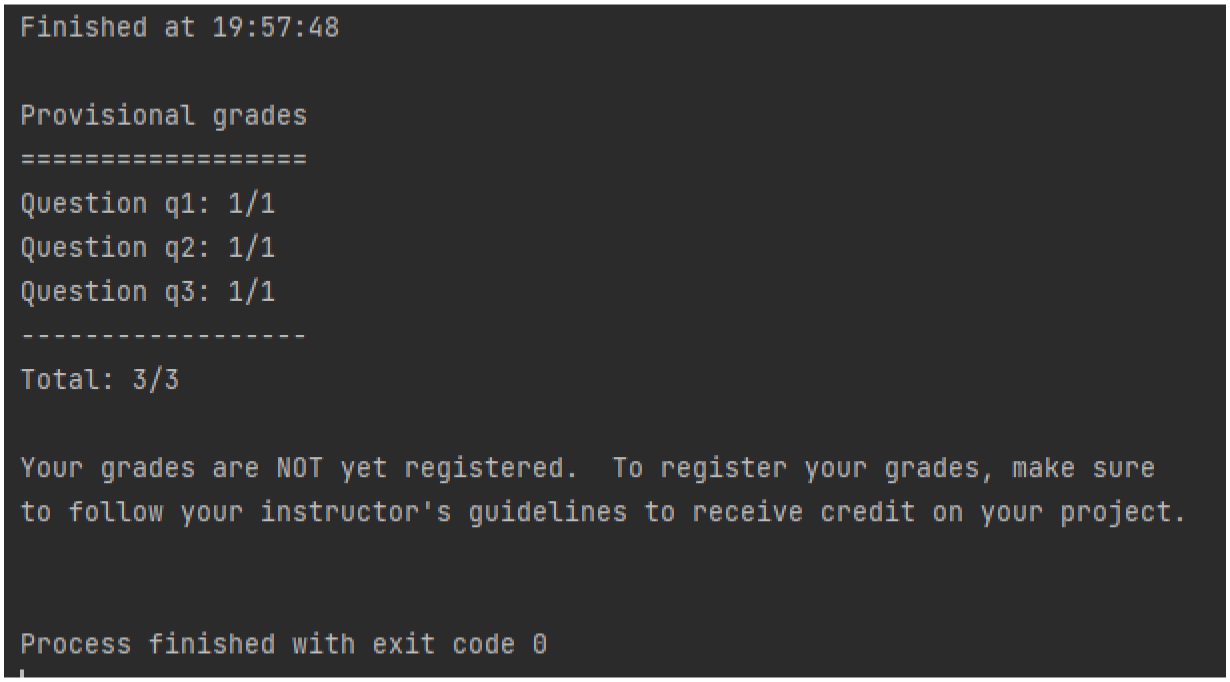
```
def shopSmart(orderList, fruitShops):
    """
        orderList: List of (fruit, numPound) tuples
        fruitShops: List of FruitShops
    """
    "*** YOUR CODE HERE ***"
    minCost = float('inf')
    bestShop = None
    for shop in fruitShops:
        cost = shop.getPriceOfOrder(orderList)
        if cost is not None and cost < minCost:
            minCost = cost
            bestShop = shop

    return bestShop
```

What I did here is first declared a minCost variable to keep track of the minimum cost and bestShop variable to keep track of the best shop based on minimum cost. So the function takes an orderList and a list of FruitShop objects. Then it iterates through the list of shops, and for each shop it calls the getPriceOfOrder method with the orderList as an argument. If the returned cost is not None and less than the current minCost, it updates the minCost variable with the new cost and the bestShop variable with the current shop. Finally, it returns the bestShop variable which is the shop that has the lowest cost for the provided orderList.

We imported Shop.py in this .py file. So I had all the necessary functions like getPriceOfOrder here.

FINAL VERDICT:



```
Finished at 19:57:48

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.



Process finished with exit code 0
```