Fall 2023 CS5368 Intelligent Systems Project 0 Report

First Name	KanvaKoushik
Last Name	Gopavarapu
Student ID	R11849414

Please answer the following questions and submit them through Blackboard. Be sure to submit it to Project 0 report submission. DO NOT write the report by hand and submit a scanned version. Just write the answers in word document and submit it. Both word and PDF submission are accepted.

1. What did Problem 1 ask for? How did you implement it?

a. Problem 1 asked for the code to return the summation of 2 numbers. I have implemented the following code print("Passed a=%s and b=%s, returning a+b=%s" % (a, b, a + b)) return a + b which returns the summation of 2 numbers. The output is as follows,

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

Passed a=1 and b=1, returning a+b=2
*** PASS: test_cases\q1\addition1.test
***     add(a,b) returns the sum of a and b
Passed a=2 and b=3, returning a+b=5
*** PASS: test_cases\q1\addition2.test
***     add(a,b) returns the sum of a and b
Passed a=10 and b=-2.1, returning a+b=7.9
*** PASS: test_cases\q1\addition3.test
***     add(a,b) returns the sum of a and b

### Question q1: 5/5 ###
```

Fig.1 Question 1 Implementation Output.

2. What did Problem 2 ask for? How did you implement it?

a. Problem 2 asked to put up a list of fruits and their cost and to print the cost or the error if the fruit is not in the list respectively by not changing the fruit price variable. Implementation of the following code

```
"""
from __future__ import print_function

fruitPrices = {'apples': 2.00, 'oranges': 1.50, 'pears': 1.75,
               'limes': 0.75, 'strawberries': 1.00}


2 usages
def buyLotsOfFruit(orderList):
    """
        orderList: List of (fruit, numPounds) tuples

    Returns cost of order
    """
    totalCost = 0.0
    for (fruit, numPounds) in orderList:
        if fruit in fruitPrices:
            totalCost += fruitPrices[fruit] * numPounds
        else:
            print(f'No fruit "{fruit}"')
            return 0.0
    return totalCost

# Main Method
if __name__ == '__main__':
    "This code runs when you invoke the script from the command line"
    orderList = [ ('apples', 2.0), ('pears', 3.0), ('limes', 4.0) ]
    print('Cost of', orderList, 'is', buyLotsOfFruit(orderList))
    orderList = [ ('apples', 2.0), ('pears', 3.0), ('limes', 4.0), ('Banana', 1.0) ]
    print('Cost of', orderList, 'is', buyLotsOfFruit(orderList))
```

Fig.2 Question 2 Implementation code.

The output for the implemented code is as follows giving us the name of the fruit along with it's price and printing 0.0 if the fruit is not present.

```
C:\Users\koush\AppData\Local\Microsoft\WindowsApps\python3.10.exe "D:\TTU\Classes\2023\Fall\Intelligent Systems\Project\0\CS5368_Project0\buyLotsOfFruit.py"
Cost of [('apples', 2.0), ('pears', 3.0), ('limes', 4.0)] is 12.25
No fruit "Banana"
Cost of [('apples', 2.0), ('pears', 3.0), ('limes', 4.0), ('Banana', 1.0)] is 0.0

Process finished with exit code 0
```

Fig.2 Question 2 Implemented output.

3. What did Problem 3 ask for? How did you implement it?

a. Problem 3 asked to create a list of fruits and return the whole order cost in total within 2 shops. By implementing the code below, the output comes as the fruit along with its price and the best shop to buy for that particular order cand its cost.

```
def shopSmart(orderList, fruitShops):
    """
        orderList: List of (fruit, numPound) tuples
        fruitShops: List of FruitShops
    """
    min_cost = 10000
    min_shop = None
    for shop in fruitShops:
        totalCost = 0
        for (fruit, numPounds) in orderList:
            if fruit in shop.fruitPrices:
                totalCost += shop.fruitPrices[fruit] * numPounds
        if min_cost > totalCost:
            min_cost = totalCost
            min_shop = shop
    return min_shop

if __name__ == '__main__':
    "This code runs when you invoke the script from the command line"
    orders = [('apples', 1.0), ('oranges', 3.0)]
    dir1 = {'apples': 2.0, 'oranges': 1.0}
    shop1 = shop.FruitShop( name: 'shop1', dir1)
    dir2 = {'apples': 1.0, 'oranges': 5.0}
    shop2 = shop.FruitShop( name: 'shop2', dir2)
    shops = [shop1, shop2]
    print("For orders ", orders, ", the best shop is", shopSmart(orders, shops).getName())
    orders = [('apples', 3.0)]
    print("For orders: ", orders, ", the best shop is", shopSmart(orders, shops).getName())
```

Fig.3 Question 3 Implemented code.

```
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
For orders  [('apples', 1.0), ('oranges', 3.0)] , the best shop is shop1
For orders:  [('apples', 3.0)] , the best shop is shop2

Process finished with exit code 0
```

Fig.4 Question 3 Implemented output.

4. What happens when you run the following codes?

python autograder.py -q q3

```
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
```

Fig.5 Python autograder-q3.

python autograder.py -t test_cases/q3/select_shop1

```
*** PASS: test_cases\q3\select_shop1.test
***     shopSmart(order, shops) selects the cheapest shop
```

Fig.6 Python autograder.py – t test_cases/q3/select shop1