Ex.No.3	DICTIONARY MANIPULATION	Reg.No: URK23CS1261
10.1.24		

3 A). Write a Python program to update the given dictionary 'inventory' by adding a 'pocket' with items 'seashell', 'strange berry', and 'lint', arranging the 'backpack' items, removing 'dagger' from 'backpack', and increasing 'gold' by 50?

inventory = {
'gold' : 500,
'pouch' : ['flint', 'twine', 'gemstone'],
'backpack' : ['xylophone', 'dagger', 'bedroll', 'bread loaf']
}

Aim: The objective of this program is to modify the key and values in a dictionary.

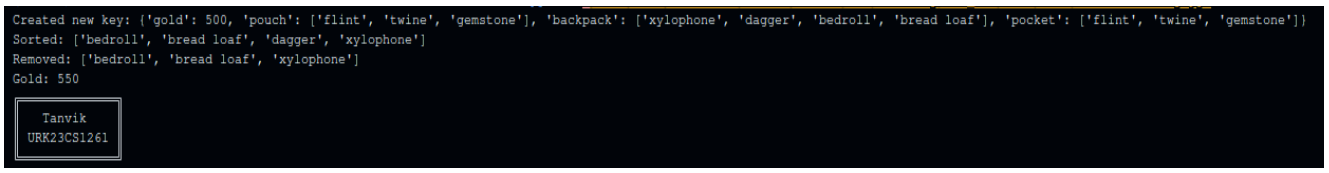
Algorithm:

```
Step 1: Start the program.
Step 2: Initialize a dictionary named 'inventory'.
Step 3: Update the 'inventory' dictionary by adding a new key 'pocket' with a list of items.
Step 4: Print the message indicating the creation of the new key.
Step 5: Sort the list of items under the 'backpack' key in the 'inventory' dictionary.
Step 6: Print the message indicating the sorting of the 'backpack' items.
Step 7: Remove the item 'dagger' from the list under the 'backpack' key in the 'inventory' dictionary.
Step 8: Print the message indicating the removal of 'dagger' from the 'backpack'.
Step 9: Retrieve the value under the 'gold' key in the 'inventory' dictionary and add 50 to it.
Step 10: Print the updated 'gold' value.
Step 11: Stop the program.
```

Program:

```
inventory = {'gold' : 500,'pouch' : ['flint', 'twine', 'gemstone'],
'backpack' : ['xylophone', 'dagger', 'bedroll', 'bread loaf']}
inventory.update({"pocket": ['flint', 'twine', 'gemstone']})
print(f"Created new key: {inventory}")
inventory.get('backpack').sort()
print(f"Sorted: {inventory.get('backpack')}")
inventory.get('backpack').remove('dagger')
print(f"Removed: {inventory.get('backpack')}")
add = inventory['gold'] + 50
print(f"Gold: {add}")
print("╔══════════════╗\n║   Tanvik    ║\n║ URK23CS1261 ║\n╚══════════════╝")
```

Output:



```
Created new key: {'gold': 500, 'pouch': ['flint', 'twine', 'gemstone'], 'backpack': ['xylophone', 'dagger', 'bedroll', 'bread loaf'], 'pocket': ['flint', 'twine', 'gemstone']}
Sorted: ['bedroll', 'bread loaf', 'dagger', 'xylophone']
Removed: ['bedroll', 'bread loaf', 'xylophone']
Gold: 550

╔═════════════╗
║   Tanvik    ║
║ URK23CS1261 ║
╚═════════════╝
```

Result: Thus, The program has successfully produced the desired output.

3 B) Write a python program to create a new dictionary named 'prices' and assign specific values to it, such as "banana": 4, "apple": 2, "orange": 1.5, and "pear": 3? Subsequently, how do you loop through the keys in 'prices', displaying each key along with its price and stock information? Lastly, how do you calculate and print the total revenue by multiplying the price and stock for each food item in the 'prices' dictionary?

Aim: The objective of this program is to create a dictionary called 'prices' with set values, display each item's price and stock, and calculate the total revenue from selling all items.

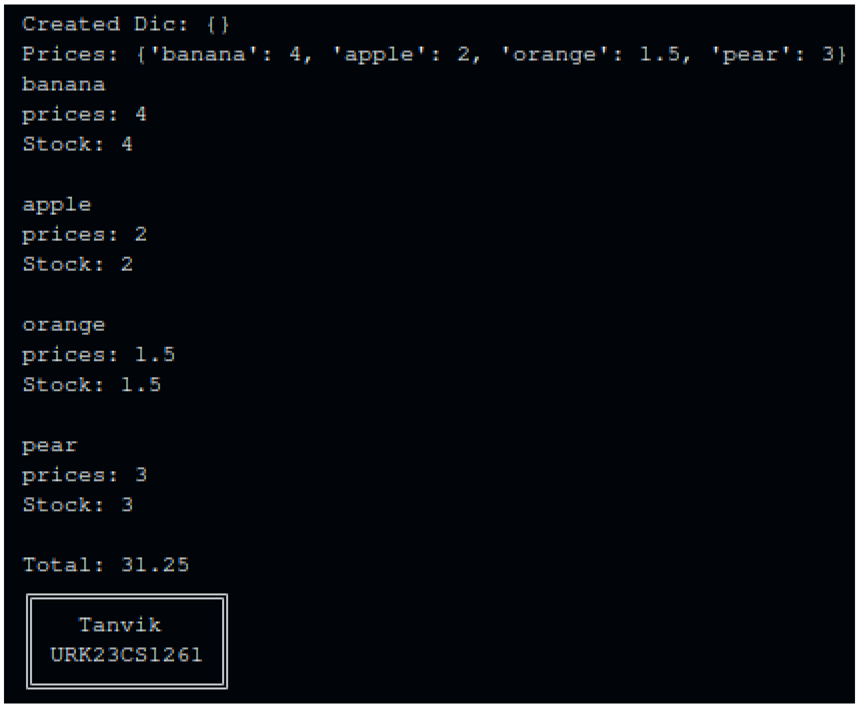
Algorithm:

```
Step 1: Start the program.
Step 2: Initialize an empty dictionary named 'prices'.
Step 3: Update the 'prices' dictionary with the given values for "banana", "apple", "orange", and "pear".
Step 4: Print the 'prices' dictionary after the update.
Step 5: Create a 'stock' dictionary with quantities for "banana", "apple", "orange", and "pear".
Step 6: Initialize a variable named 'total' and set it to zero.
Step 7: Iterate the 'prices' dictionary using a for loop, with 'key' as the item name and 'value' as its price.
Step 8: Nested inside the first loop, iterate through the 'stock' dictionary to find the corresponding stock quantity for each item.
Step 9: If the 'key' matches the 'stock_name', calculate the revenue for the item by multiplying its price ('value') with the stock quantity ('stock_value').
Step 10: Print the item name, its price, and stock quantity.
Step 11: Add the calculated revenue to the 'total' variable, print the total revenue, signature at the end.
Step 12: Stop the program..
```

Program:

```
prices = {}
print(f"Created Dic: {prices}")
prices.update({"banana": 4,
              "apple": 2,
              "orange": 1.5,
              "pear": 3
})
print(f"Prices: {prices}")
Stock = {
      "banana": 6,
      "apple": 0,
      "orange": 32,
      "pear": 15}
total = 0
for key, value in prices.items():
  for stock_name, stock_value in prices.items():
    if key == stock_name:
      total += value * stock_value
      print(f"{key}\nprices: {value}\nStock: {stock_value}\n")
print(f"Total: {total}")
print("╔══════════════╗\n║   Tanvik   ║\n║ URK23CS1261 ║\n╚══════════════╝")
```

Output:



```
Created Dic: {}
Prices: {'banana': 4, 'apple': 2, 'orange': 1.5, 'pear': 3}
banana
prices: 4
Stock: 4

apple
prices: 2
Stock: 2

orange
prices: 1.5
Stock: 1.5

pear
prices: 3
Stock: 3

Total: 31.25
```

```
 Tanvik
URK23CS1261
```

Result: Thus, The program has successfully produced the desired output.

3 C) Develop a Python program to create a list 'groceries' with values "banana", "orange", and "apple"? Additionally, how do you define two dictionaries 'stock' and 'prices' with quantities and prices for each item? Lastly, how do you implement a function named 'compute_bill' that calculates the total cost for items in a given list, considering the stock count and subtracting one from the count if the item is in stock?

stock = { "banana": 6, "apple": 0, "orange": 32, "pear": 15 }
prices = { "banana": 4, "apple": 2, "orange": 1.5, "pear": 3 }

Aim: The objective of this program is to create a list 'groceries' with specific items, define dictionaries 'stock' and 'prices' with quantities and prices, and implement a function 'compute_bill' to calculate the total cost while considering stock availability.

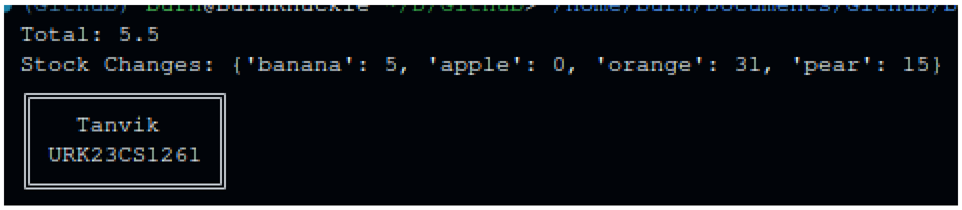
Algorithm:

```
Step 1: Start the program.
Step 2: Initialize a list named 'groceries' with values "banana", "orange", and "apple".
Step 3: Initialize a dictionary named 'stock', 'prices'.
Step 4: Define a function named 'compute_bill' that takes a list 'food' as input.
Step 5: Inside 'compute_bill', initialize a variable 'total' and set it to zero.
Step 6: Use a for loop to iterate through each item 'name' in the 'food' list.
Step 7: Nested inside the first loop, use another for loop to iterate through 'stock' to find the corresponding stock quantity for each item.
Step 8: If 'name' is present in 'prices' and 'name' matches 'stock_name', check if the stock quantity for the item is greater than zero.
Step 9: If the stock is available, decrement the stock count by 1 and add the price of the item to 'total'.
Step 10: Print the item name, its price, and stock quantity.
Step 11: Add the calculated revenue to the 'total' variable, print the total revenue, the signature.
Step 12: Call the 'compute_bill' function with the 'groceries' list and print the changes in stock.
Step 13: Stop the program.
```

Program:

```
groceries = ["banana", "orange","apple"]
stock = {"banana": 6,"apple": 0,"orange": 32,"pear": 15}
prices = {"banana": 4,"apple": 2,"orange": 1.5,"pear": 3}
def compute_bill(food):
    total = 0
    for name in food:
        for stock_name, stock_value in stock.items():
            if name in prices and name == stock_name:
                if stock[name] > 0:
                    stock[name] -= 1
                    total += prices.get(name)
    return total
print(f"Total: {compute_bill(groceries)}\nStock Changes: {stock}")
print("╔══════════════════╗\n║   Tanvik   ║\n║ URK23CS1261 ║\n╚══════════════════╝")
```

Output:



```
Total: 5.5
Stock Changes: {'banana': 5, 'apple': 0, 'orange': 31, 'pear': 15}

  Tanvik
URK23CS1261
```

Result: Thus, The program has successfully produced the desired output.

3 D) Write a Python program to create a gradebook for a teacher's students? Include steps like initializing student dictionaries, populating them with data, and creating a list of students. Additionally, describe how to print each student's information, calculate average scores, assign letter grades, and determine the class average.

Aim: The objective of this program is to set up a student gradebook for a teacher, including creating student dictionaries, calculating individual and class averages, and assigning letter grades based on weighted scores.

Algorithm:

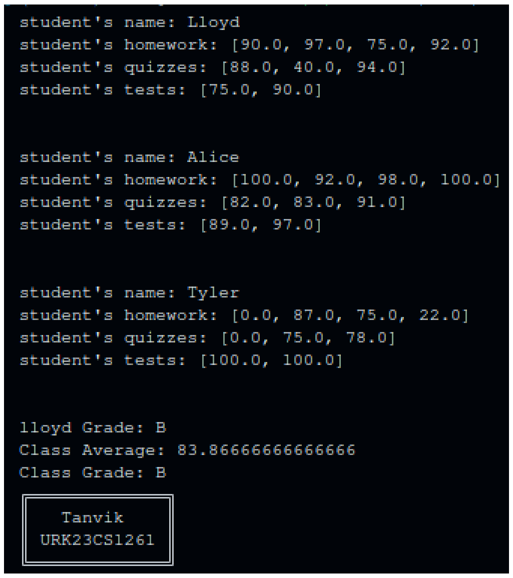
Step 1: Start the program.
Step 2: Create a list 'students' containing student dictionaries.
Step 3: Iterate through each student in the 'students' list.
Step 4: Nested inside the loop, iterate through the key-value pairs of each student's dictionary and print the information.
Step 5: Move to the next line after printing each student's details.
Step 6: Define a function 'average' that takes a list of numbers, calculates the total, converts it to a float, and returns the average.
Step 7: Define a function 'get_average' that takes a student dictionary as input, calculates average scores for homework, quizzes, and tests, applies weights (10%, 30%, 60%), and returns the overall average.
Step 8: Define a function 'get_letter_grade' that assigns a letter grade based on the numeric score provided.
Step 9: Calculate the letter grade for Lloyd's average using 'get_average' and 'get_letter_grade', then print the result.
Step 10: Define a function 'get_class_average' that takes a list of students, calculates the individual averages for each student using 'get_average', and returns the class average.
Step 11: Calculate the class average and letter grade using 'get_class_average', 'get_letter_grade', and print the results.
Step 12: End the program..

Program:

```
lloyd = {"name": "Lloyd","homework": [90.0,97.0,75.0,92.0],"quizzes": [88.0,40.0,94.0],
"tests": [75.0,90.0]}
alice = {"name": "Alice","homework": [100.0, 92.0, 98.0, 100.0],"quizzes": [82.0, 83.0, 91.0],
"tests": [89.0, 97.0]}
tyler = {"name": "Tyler","homework": [0.0, 87.0, 75.0, 22.0],"quizzes": [0.0, 75.0, 78.0],
"tests": [100.0, 100.0]}
students = [lloyd, alice, tyler]
for student in students:
    for key, value in student.items():
        print(f"student's {key}: {value}")
    print('\n')
def average(num_list):
    total = sum(num_list)
    total = float(total/len(num_list)) # for extra check : if len(num_list)> 0 else 0)
    return total
```

```
def get_average(student):
   homework = average(student['homework'])
   quizzes = average(student['quizzes'])
   tests = average(student['tests'])
   return (homework * 0.1) + (quizzes * 0.3) + (tests * 0.6)
def get_letter_grade(score: int):
   if score >= 90:
      return 'A'
   elif score >= 80:
      return 'B'
   elif score >= 70:
      return 'C'
   elif score >= 60:
      return 'D'
   else:
      return 'F'
result = get_letter_grade(get_average(lloyd))
print(f"lloyd Grade: {result}")
def get_class_average(students):
   results = []
   for student in students:
      results.append(get_average(student))
   return average(results)
class_avg = get_class_average(students)
class_grade = get_letter_grade(class_avg)
print(f"Class Average: {class_avg}\nClass Grade: {class_grade}")
print("╔═══════════════╗\n║   Tanvik    ║\n║ URK23CS1261 ║\n╚═══════════════╝")
```

Output:



```
student's name: Lloyd
student's homework: [90.0, 97.0, 75.0, 92.0]
student's quizzes: [88.0, 40.0, 94.0]
student's tests: [75.0, 90.0]


student's name: Alice
student's homework: [100.0, 92.0, 98.0, 100.0]
student's quizzes: [82.0, 83.0, 91.0]
student's tests: [89.0, 97.0]


student's name: Tyler
student's homework: [0.0, 87.0, 75.0, 22.0]
student's quizzes: [0.0, 75.0, 78.0]
student's tests: [100.0, 100.0]


lloyd Grade: B
Class Average: 83.86666666666666
Class Grade: B
╔═════════════╗
║   Tanvik    ║
║ URK23CS1261 ║
╚═════════════╝
```

Result: Thus, The program has successfully produced the desired output.