ASSIGNMENT-3

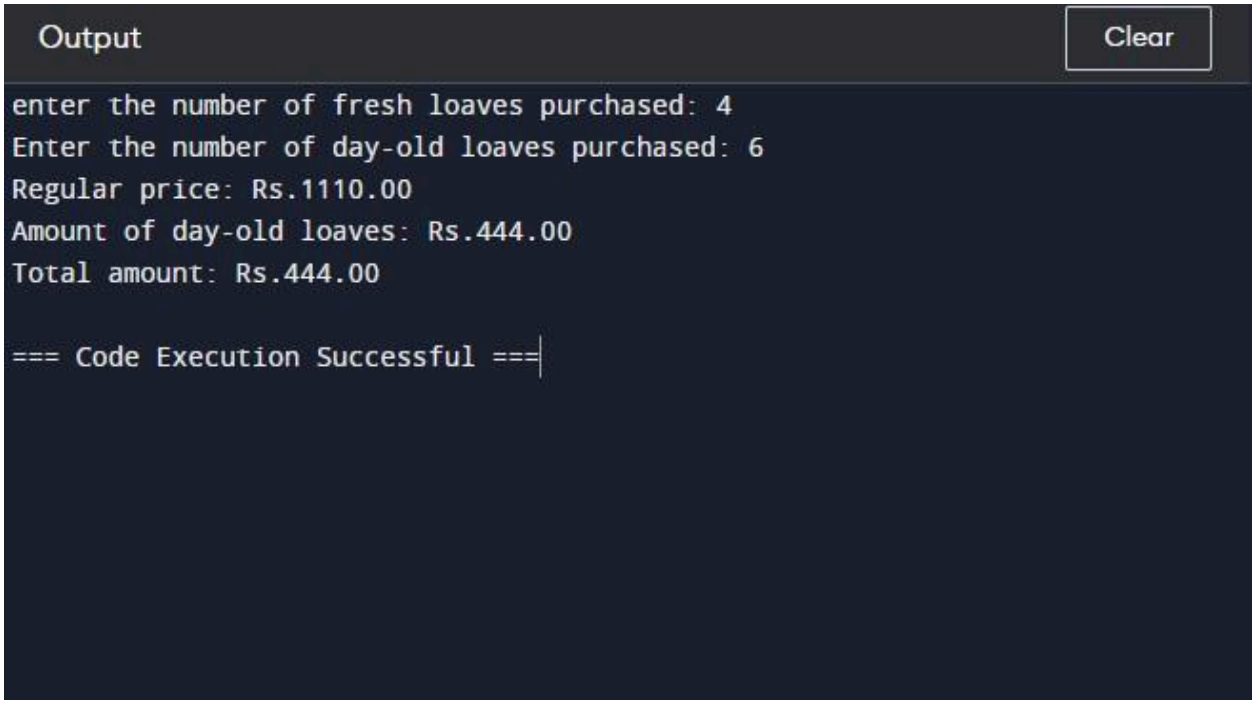
1. A bakery sells loaves of bread for 185 rupees each. Day old bread is discounted by 60 percent. Write a python program that begins by reading the number of loaves of day old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price. All of the values should be displayed using two decimal places, and the decimal points in all of the numbers should be aligned when reasonable values are entered by the user.

Sample Input: Enter the number of fresh loaves purchased: 5
Enter the number of day-old loaves purchased: 3

Sample Output: Regular price: Rs.185.00
Amount of new loaves: 925.00
Amount of day-old loaves: 333.00
Total amount: Rs. 1258.00
Test cases: 1. 4, 6

Python Code for the question:

```
fresh_loaves=int(input(“enter the number of fresh loaves purchased: “))
num_day_old_loaves = int(input("Enter the number of day-old loaves purchased: "))

regular_price = 185 * num_day_old_loaves
discount = regular_price * 0.6

total_price_day_old = regular_price - discount


print("Regular price: Rs.{:.2f}".format(regular_price))
print("Amount of day-old loaves: Rs.{:.2f}".format(total_price_day_old))
print("Total amount: Rs.{:.2f}".format(total_price_day_old))
```



2. Given two strings “s” and “t”, determine if they are isomorphic. Two strings “s” and “t” are isomorphic if the characters in “s” can be replaced to get “t”. All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

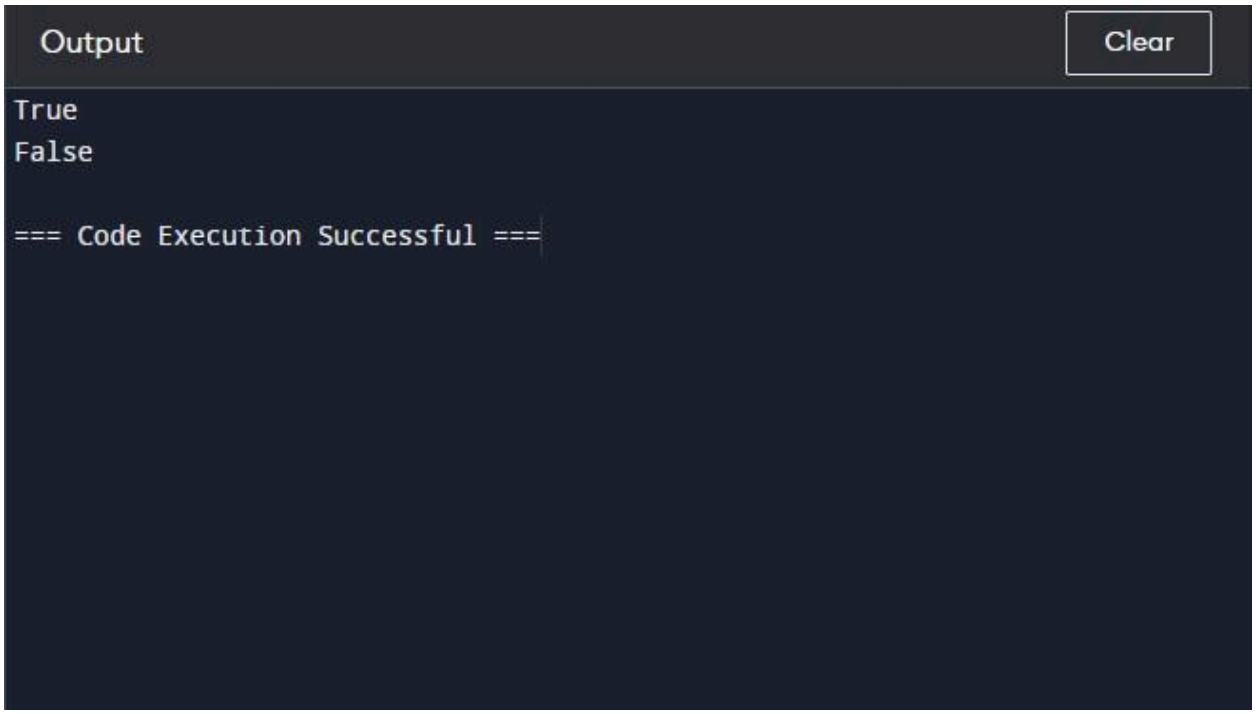
Constraints: ✓ s and t consist of any valid ascii character.

Test Cases: 1.Input: s = "egg", t = "add" Output: true
2.Input: s = "foo", t = "bar" Output: false
3.Input: s = "paper", t = "title" Output: true
4.Input: s = "fry", t = "sky" Output: true
5. Input: s = "apples", t = "apple" Output: false

Python Code for the question:

```
def is_isomorphic(s: str, t: str) -> bool:
    if len(s) != len(t):
        return False

    s_to_t = {}
```

```
    t_to_s = {}

    for c_s, c_t in zip(s, t):
        if c_s not in s_to_t and c_t not in t_to_s:
            s_to_t[c_s] = c_t
            t_to_s[c_t] = c_s
        elif s_to_t.get(c_s) != c_t or t_to_s.get(c_t) != c_s:
            return False
    return True

print(is_isomorphic("egg", "add"))
print(is_isomorphic("foo", "bar"))
```



3. Given n non-negative integers a1, a2,a3,…and where each represents a point at coordinate (i, ai) . ‘ n ‘ vertical lines are drawn such that the two endpoints of line i are at (i, ai) and (i,0). Find two lines, which together with x-axis forms a container, such that the container contains the most water. The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume sounds weird but this is the 2D plane we are working with for simplicity).

Note: You may not slant the container

.
Test case: 1.Input: array = [1, 5, 4, 3] Output: 6
2.Input: array = [3, 1, 2, 4, 5] Output: 12
3.Input: array = [1,8,6,2,5,4,8,3,7] Output: 49
4.Input: array = [1,1] Output: 1
5.Input: array = [7,3] Output: 3

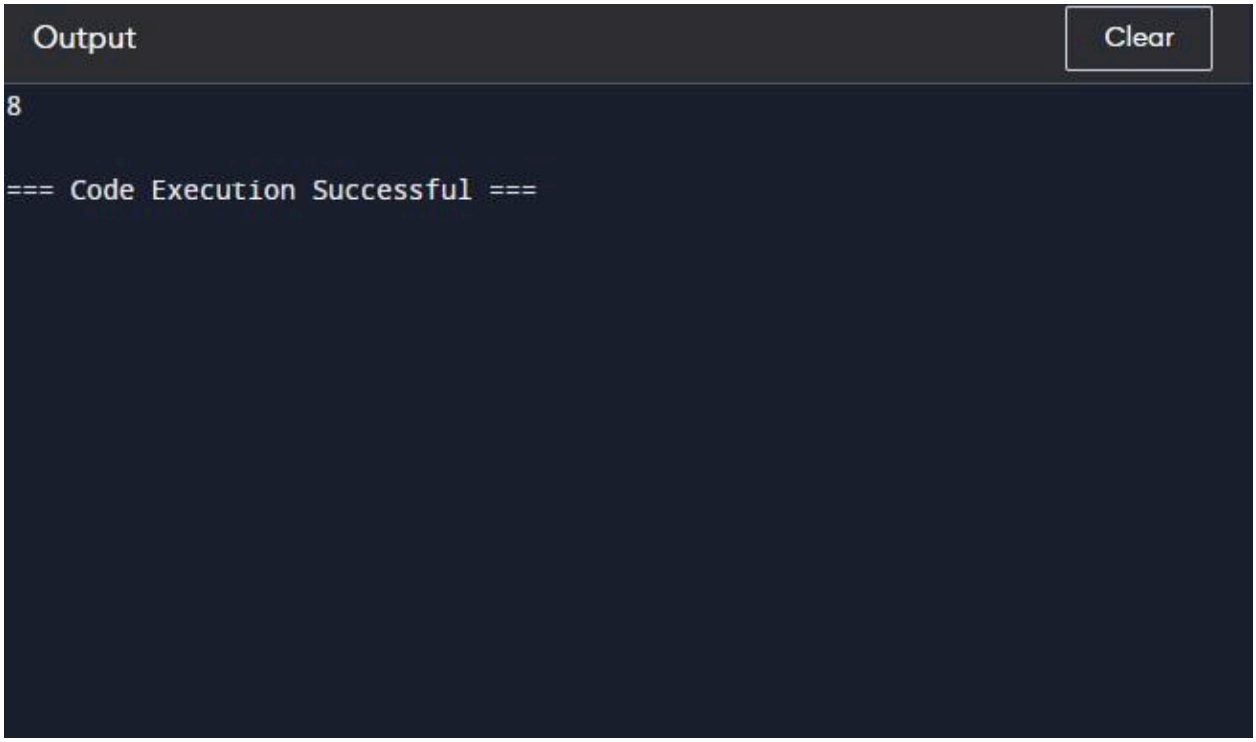
Python Code for the question:

```
def max_area(height):
    left = 0
    right = len(height) - 1
    max_area = 0

    while left < right:
        width = right - left
        height_min = min(height[left], height[right])
        area = width * height_min
        max_area = max(max_area, area)

        if height[left] < height[right]:
            left += 1
        else:
            right -= 1
    return max_area

print(max_area([1,5,4,3])
print(max_area([3, 1, 2, 4, 5]))
print(max_area([1,8,6,2,5,4,8,3,7]))
```

4. You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?
Test Case: 1.Input: n = 2 Output: 2
2.Input: n = 3 Output: 3
3.Input: n = 4 Output: 5
4.Input: n = 1 Output: 1
5.Input: n = 5 Output: 8

Python Code for the question:

```
def climb_stairs(n: int) -> int:
    if n == 1:
        return 1
    dp = [0] * (n + 1)
    dp[1] = 1
    dp[2] = 2
    for i in range(3, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[n]

print(climb_stairs(5))
```



5. In daily share trading, a buyer buys shares in the morning and sells them on the same day. If the trader is allowed to make at most 2 transactions in a day, whereas the second transaction can only start after the first one is complete (Buy->sell->Buy->sell). Given stock prices throughout the day, find out the maximum profit that a share trader could have made.

Test Case: 1.Input: prices = [7,1,5,3,6,4] Output: 7
2.Input: prices = [7,6,4,3,1] Output: 0
3.Input: [10, 22, 5, 75, 65, 80] Output:87
4.Input: [2, 30, 15, 10, 8, 25, 80] Output:100
5. Input: [5,25,3,10,7,9] Output:27

Python Code for the question:

```
def max_profit(prices):
    if not prices:
        return 0

    buy1, sell1, buy2, sell2 = float('-inf'), 0, float('-inf'), 0

    for price in prices:
        buy1 = max(buy1, -price)
        sell1 = max(sell1, buy1 + price)
        buy2 = max(buy2, sell1 - price)
```

```
        sell2 = max(sell2, buy2 + price)
    return sell2

print(max_profit([7,6,4,3,1]))
print(max_profit([10, 22, 5, 75, 65, 80]))
```