SAVEETHA SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTERSCIENCE AND ENGINEERING

CSA0889 – Python Programming

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.

SOURCE CODE:

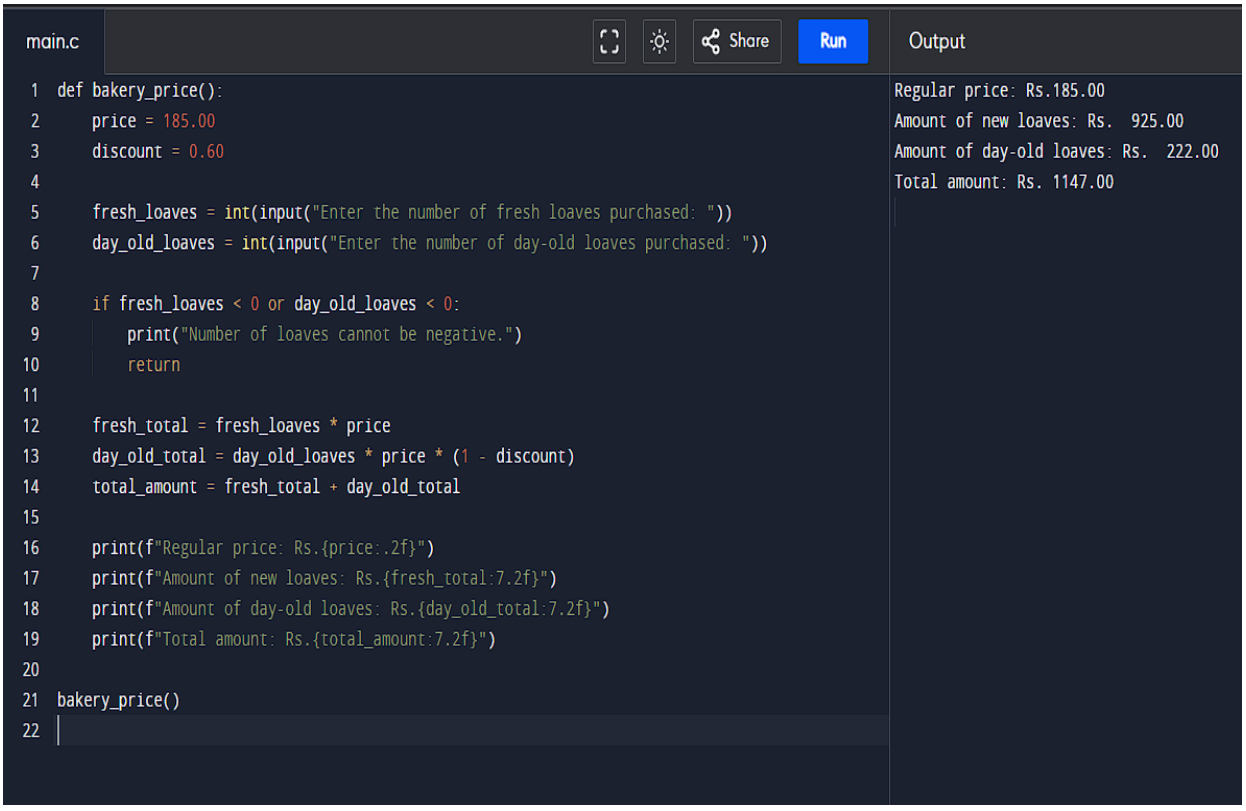
```
def bakery_price():
    price = 185.00
    discount = 0.60
    fresh_loaves = int(input("Enter the number of fresh loaves purchased: "))
    day_old_loaves = int(input("Enter the number of day-old loaves purchased:"))

    if fresh_loaves < 0 or day_old_loaves < 0:
        print("Number of loaves cannot be negative.")
        return
    fresh_total = fresh_loaves * price
    day_old_total = day_old_loaves * price * (1 - discount)
```

```
    total_amount = fresh_total + day_old_total

    print(f"Regular price: Rs.{price:7.2f}")
    print(f"Amount of new loaves: Rs.{fresh_total:7.2f}")
    print(f"Amount of day-old loaves: Rs.{day_old_total:7.2f}")
    print(f"Total amount: Rs.{total_amount:7.2f}")
bakery_price()
```

OUTPUT:

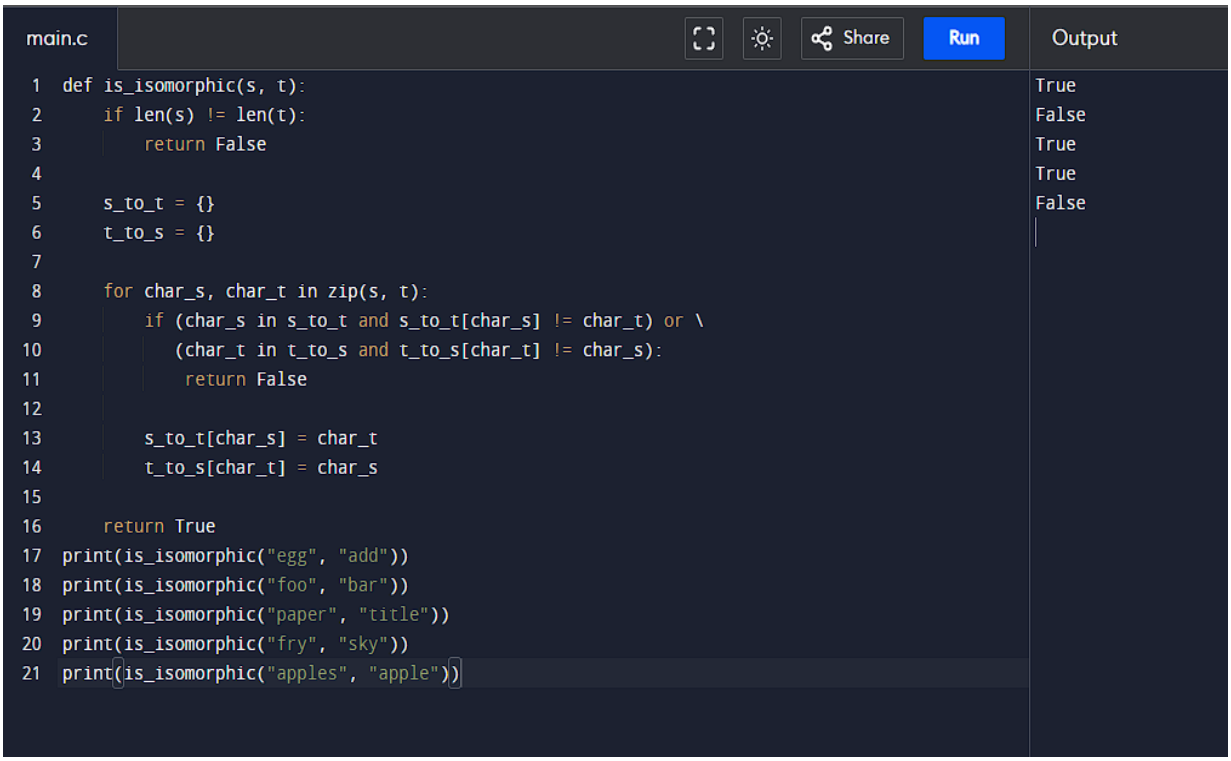


2.Given two strings “s” and “t”, determine if they are isomorphic. Two strings “s” and “t” are isomorphic if the characters in “s” can be replaced to get “t”. All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to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

SOURCE CODE:

```
def is_isomorphic(s, t):
    if len(s) != len(t):
        return False
    s_to_t = {}
    t_to_s = {}
    for char_s, char_t in zip(s, t):
        if (char_s in s_to_t and s_to_t[char_s] != char_t) or \
           (char_t in t_to_s and t_to_s[char_t] != char_s):
            return False
        s_to_t[char_s] = char_t
        t_to_s[char_t] = char_s
    return True
print(is_isomorphic("egg", "add"))
print(is_isomorphic("foo", "bar"))
print(is_isomorphic("paper", "title"))
print(is_isomorphic("fry", "sky"))
print(is_isomorphic("apples", "apple"))
```

OUTPUT:

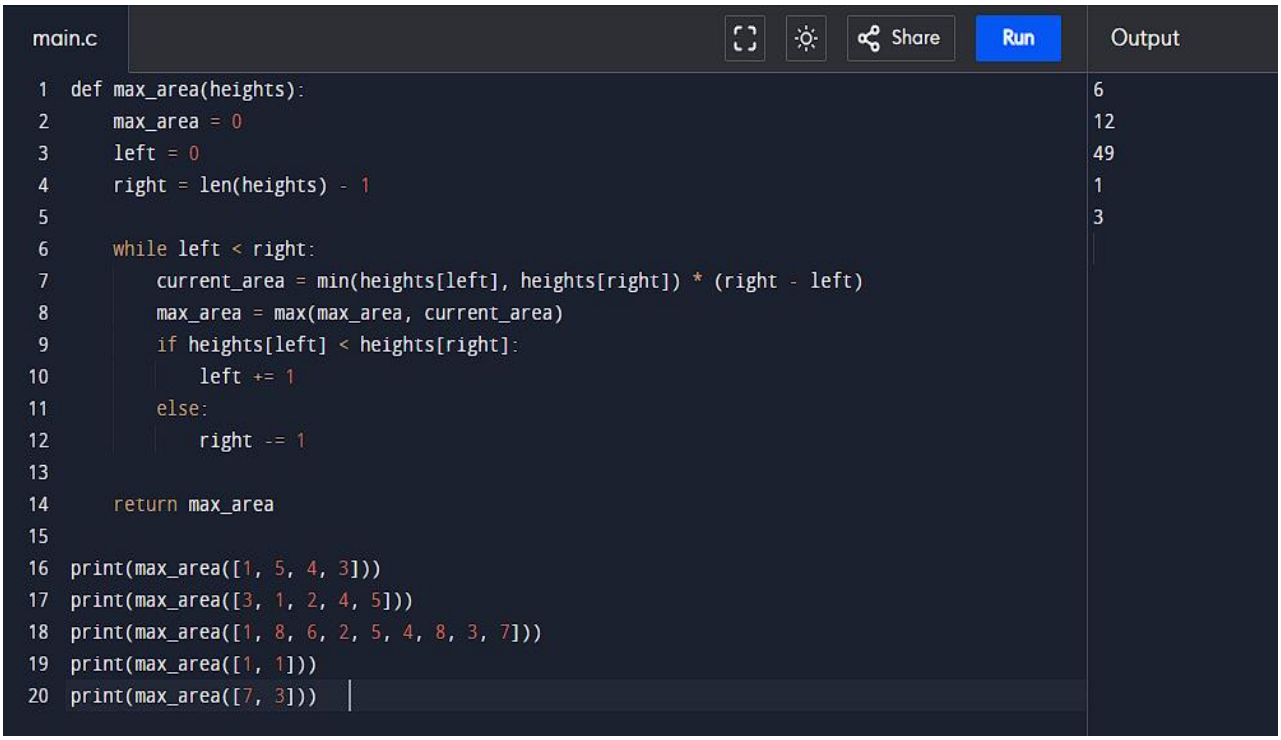


3. Given n non-negative integers a1, a2,a3,…an where each represents a point at coordinate (i, ai) . ‘ n ‘ vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i,0). Find two lines, which together with x-axis forms a container, such that the container contains the most water. 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

SOURCE CODE:

```
def max_area(heights):
 max_area = 0
    left = 0
    right = len(heights) - 1
    while left < right:
        current_area = min(heights[left], heights[right]) * (right - left)
        max_area = max(max_area, current_area)
        if heights[left] < heights[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]))
print(max_area([1, 1]))
print(max_area([7, 3]))
```

OUTPUT:



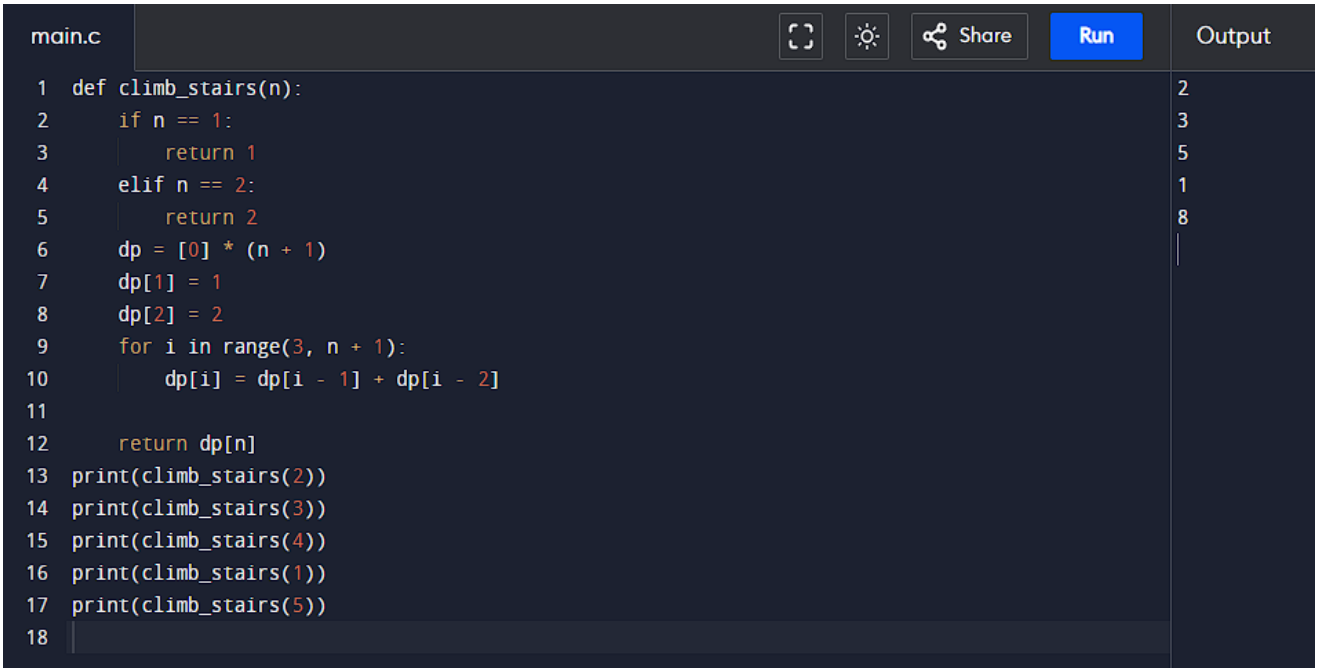
4.You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top? Test Case: 1.Input: n = 2 Output: 2 2.Input: n = 3 Output: 3 3.Input: n = 4 Output: 5 4.Input: n = 1 Output: 1 5.Input: n = 5 Output: 8

SOURCE CODE:

```
def climb_stairs(n):
    if n == 1:
        return 1
    elif n == 2:
        return 2
```

```
    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(2))
print(climb_stairs(3))
print(climb_stairs(4))
print(climb_stairs(1))
print(climb_stairs(5))
```

OUTPUT:



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

SOURCE CODE:

```
def max_profit(prices):
    if not prices:
        return 0
    first_buy = float('-inf')
    first_sell = 0
    second_buy = float('-inf')
    second_sell = 0

    for price in prices:
        first_buy = max(first_buy, -price)
        first_sell = max(first_sell, first_buy + price)
        second_buy = max(second_buy, first_sell - price)
        second_sell = max(second_sell, second_buy + price)
```

```
    return second_sell
print(max_profit([7, 1, 5, 3, 6, 4]))
print(max_profit([7, 6, 4, 3, 1]))  print(max_profit([10, 22, 5, 75, 65, 80]))
print(max_profit([2, 30, 15, 10, 8, 25, 80]))
print(max_profit([5, 25, 3, 10, 7, 9]))
```

OUTPUT: