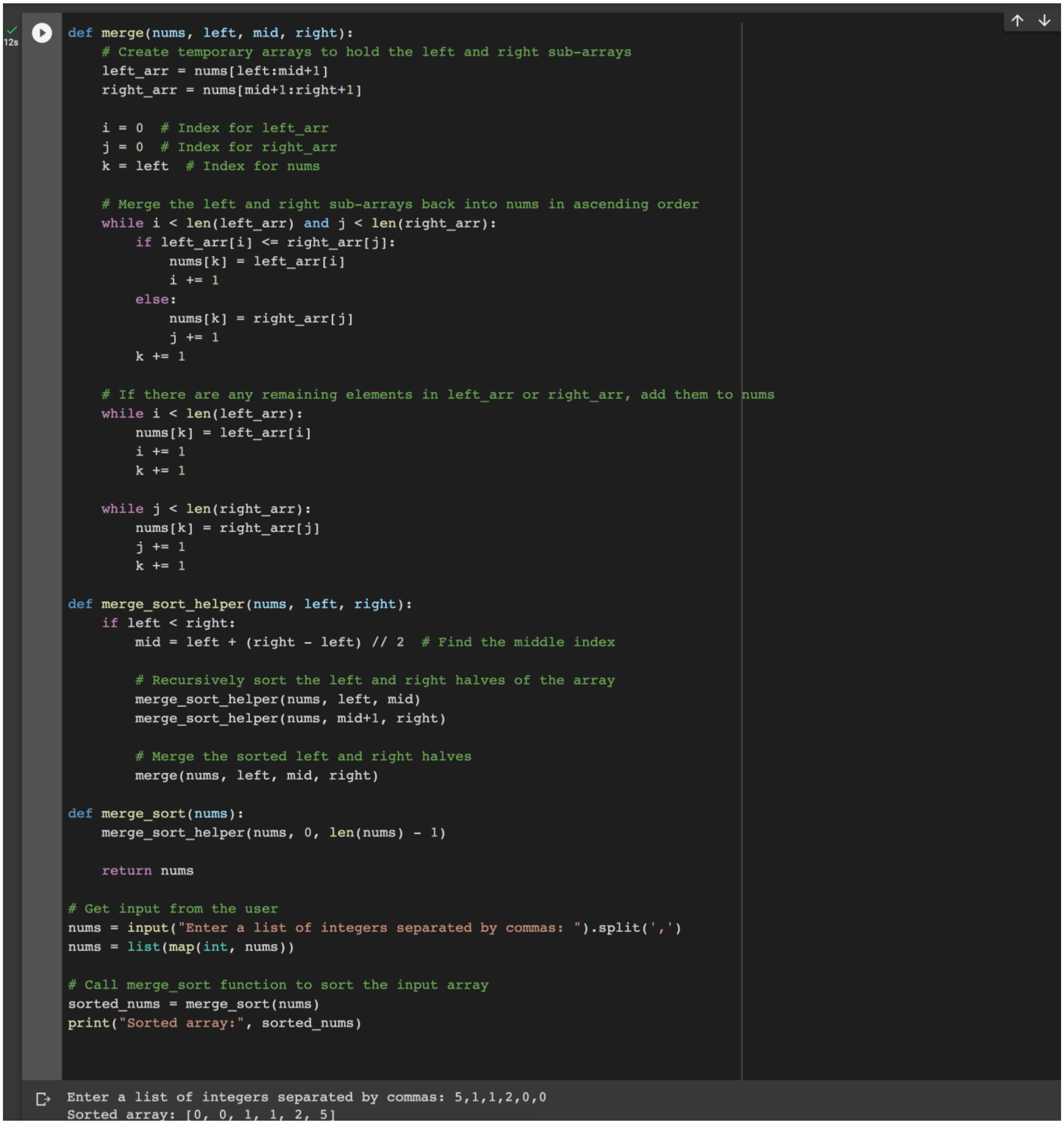
Yakshita Rakholiya
Dhruv Ranpariya
Suraj Salunkhe
Course: CS-608-21141
Assignment-4
Team-2

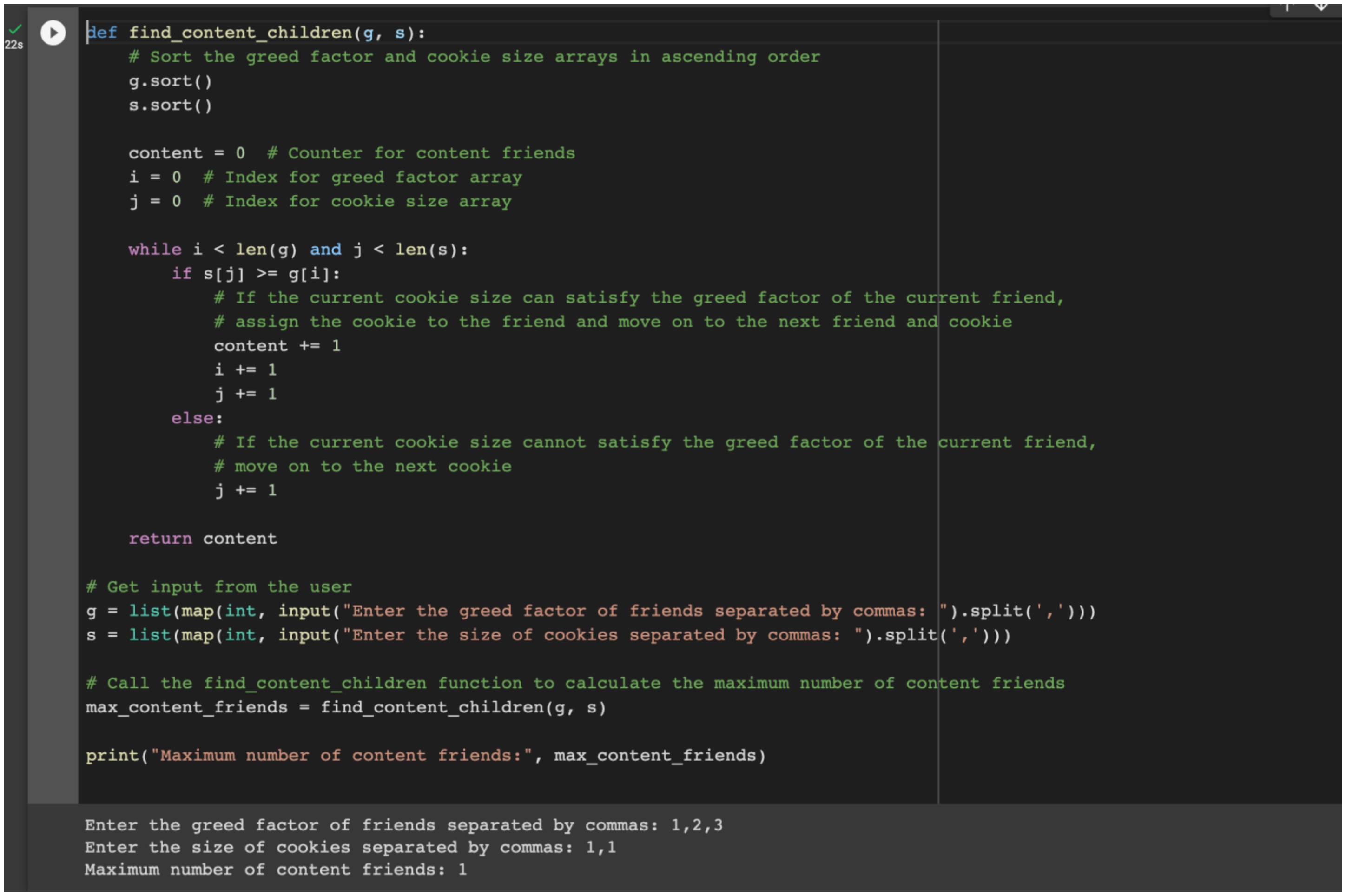
Q-1



```python
def merge(nums, left, mid, right):
    # Create temporary arrays to hold the left and right sub-arrays
    left_arr = nums[left:mid+1]
    right_arr = nums[mid+1:right+1]

    i = 0  # Index for left_arr
    j = 0  # Index for right_arr
    k = left  # Index for nums

    # Merge the left and right sub-arrays back into nums in ascending order
    while i < len(left_arr) and j < len(right_arr):
        if left_arr[i] <= right_arr[j]:
            nums[k] = left_arr[i]
            i += 1
        else:
            nums[k] = right_arr[j]
            j += 1
        k += 1

    # If there are any remaining elements in left_arr or right_arr, add them to nums
    while i < len(left_arr):
        nums[k] = left_arr[i]
        i += 1
        k += 1

    while j < len(right_arr):
        nums[k] = right_arr[j]
        j += 1
        k += 1

def merge_sort_helper(nums, left, right):
    if left < right:
        mid = left + (right - left) // 2  # Find the middle index

        # Recursively sort the left and right halves of the array
        merge_sort_helper(nums, left, mid)
        merge_sort_helper(nums, mid+1, right)

        # Merge the sorted left and right halves
        merge(nums, left, mid, right)

def merge_sort(nums):
    merge_sort_helper(nums, 0, len(nums) - 1)

    return nums

# Get input from the user
nums = input("Enter a list of integers separated by commas: ").split(',')
nums = list(map(int, nums))

# Call merge_sort function to sort the input array
sorted_nums = merge_sort(nums)
print("Sorted array:", sorted_nums)
```

```
Enter a list of integers separated by commas: 5,1,1,2,0,0
Sorted array: [0, 0, 1, 1, 2, 5]
```

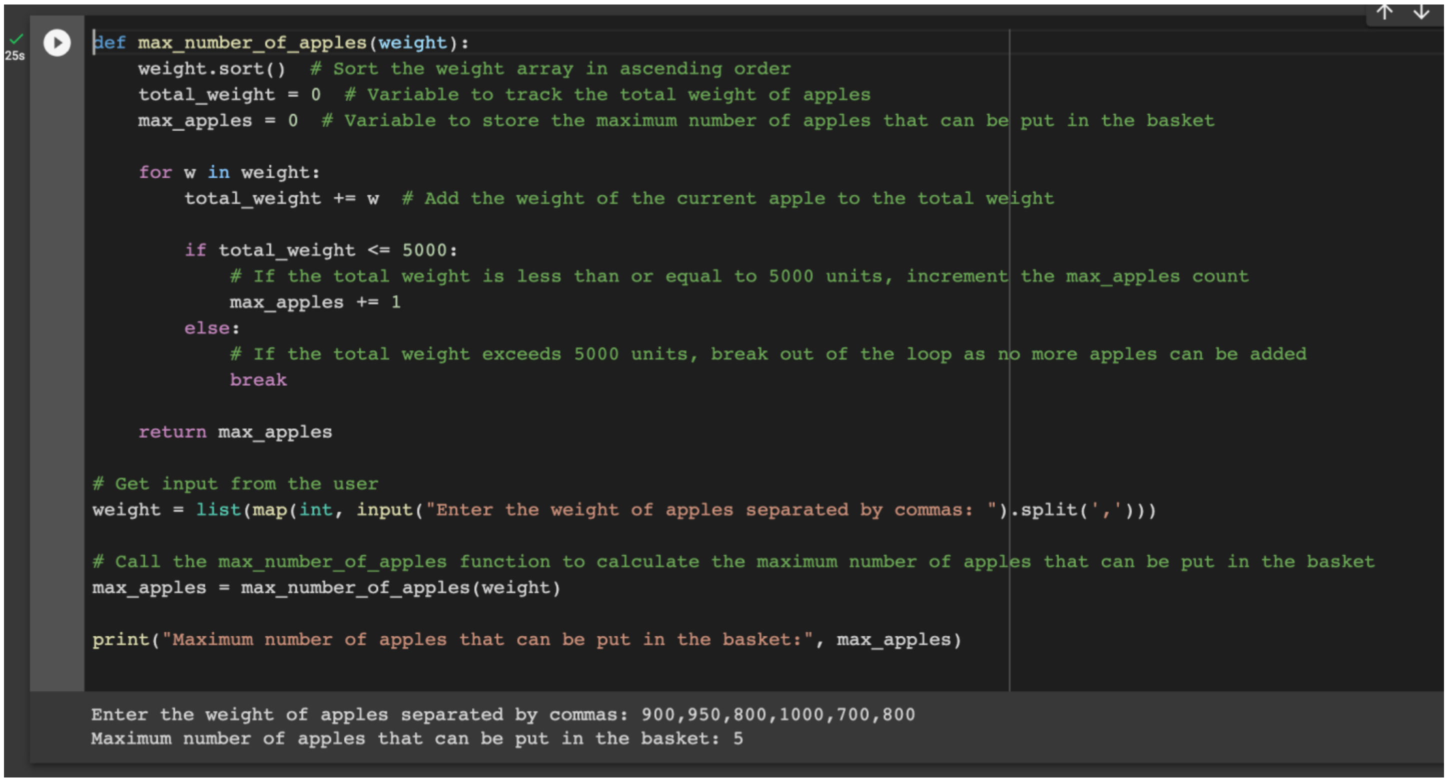
Q-2



```
def find_content_children(g, s):
    # Sort the greed factor and cookie size arrays in ascending order
    g.sort()
    s.sort()

    content = 0  # Counter for content friends
    i = 0  # Index for greed factor array
    j = 0  # Index for cookie size array

    while i < len(g) and j < len(s):
        if s[j] >= g[i]:
            # If the current cookie size can satisfy the greed factor of the current friend,
            # assign the cookie to the friend and move on to the next friend and cookie
            content += 1
            i += 1
            j += 1
        else:
            # If the current cookie size cannot satisfy the greed factor of the current friend,
            # move on to the next cookie
            j += 1

    return content

# Get input from the user
g = list(map(int, input("Enter the greed factor of friends separated by commas: ").split(',')))
s = list(map(int, input("Enter the size of cookies separated by commas: ").split(',')))

# Call the find_content_children function to calculate the maximum number of content friends
max_content_friends = find_content_children(g, s)

print("Maximum number of content friends:", max_content_friends)
```

```
Enter the greed factor of friends separated by commas: 1,2,3
Enter the size of cookies separated by commas: 1,1
Maximum number of content friends: 1
```

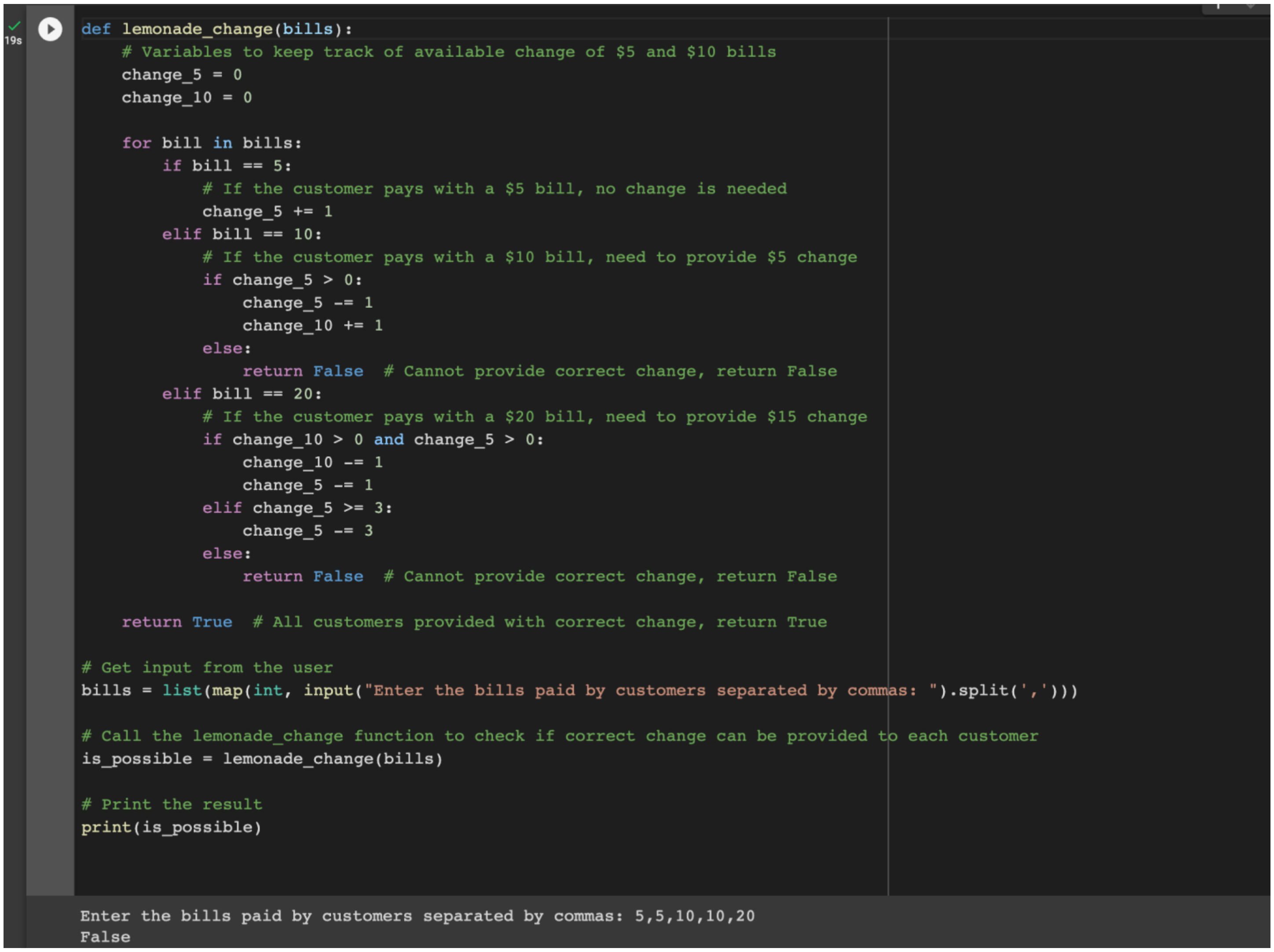
Q-3



```
def max_number_of_apples(weight):
    weight.sort()  # Sort the weight array in ascending order
    total_weight = 0  # Variable to track the total weight of apples
    max_apples = 0  # Variable to store the maximum number of apples that can be put in the basket

    for w in weight:
        total_weight += w  # Add the weight of the current apple to the total weight

        if total_weight <= 5000:
            # If the total weight is less than or equal to 5000 units, increment the max_apples count
            max_apples += 1
        else:
            # If the total weight exceeds 5000 units, break out of the loop as no more apples can be added
            break

    return max_apples

# Get input from the user
weight = list(map(int, input("Enter the weight of apples separated by commas: ").split(',')))

# Call the max_number_of_apples function to calculate the maximum number of apples that can be put in the basket
max_apples = max_number_of_apples(weight)

print("Maximum number of apples that can be put in the basket:", max_apples)
```

```
Enter the weight of apples separated by commas: 900,950,800,1000,700,800
Maximum number of apples that can be put in the basket: 5
```

Q-4



```
def lemonade_change(bills):
    # Variables to keep track of available change of $5 and $10 bills
    change_5 = 0
    change_10 = 0

    for bill in bills:
        if bill == 5:
            # If the customer pays with a $5 bill, no change is needed
            change_5 += 1
        elif bill == 10:
            # If the customer pays with a $10 bill, need to provide $5 change
            if change_5 > 0:
                change_5 -= 1
                change_10 += 1
            else:
                return False  # Cannot provide correct change, return False
        elif bill == 20:
            # If the customer pays with a $20 bill, need to provide $15 change
            if change_10 > 0 and change_5 > 0:
                change_10 -= 1
                change_5 -= 1
            elif change_5 >= 3:
                change_5 -= 3
            else:
                return False  # Cannot provide correct change, return False

    return True  # All customers provided with correct change, return True

# Get input from the user
bills = list(map(int, input("Enter the bills paid by customers separated by commas: ").split(',')))

# Call the lemonade_change function to check if correct change can be provided to each customer
is_possible = lemonade_change(bills)

# Print the result
print(is_possible)
```

```
Enter the bills paid by customers separated by commas: 5,5,10,10,20
False
```