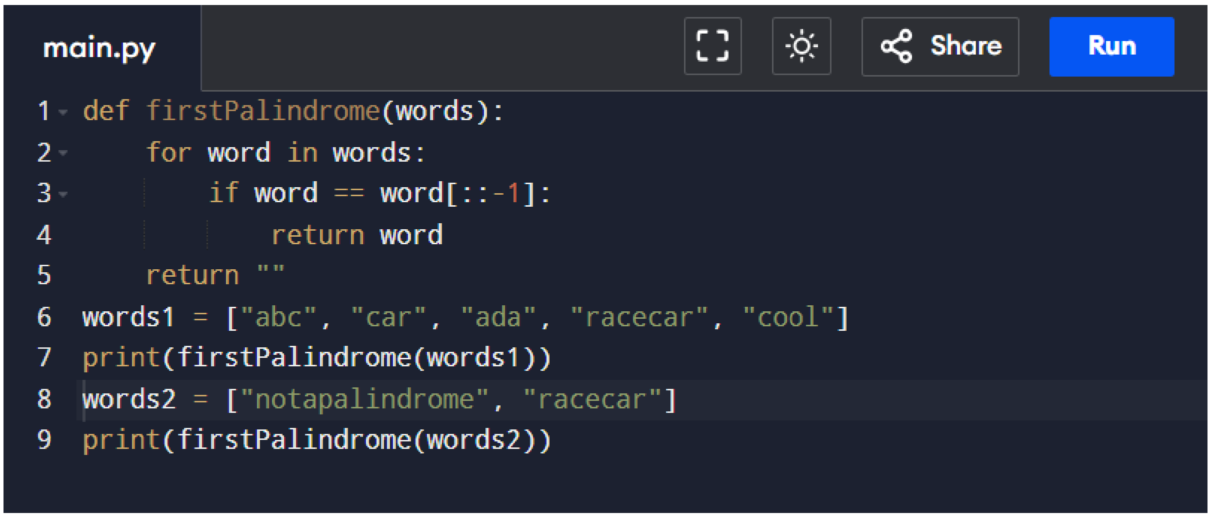
DAA LAB EXERCISE

TOPIC 1 : INTRODUCTION

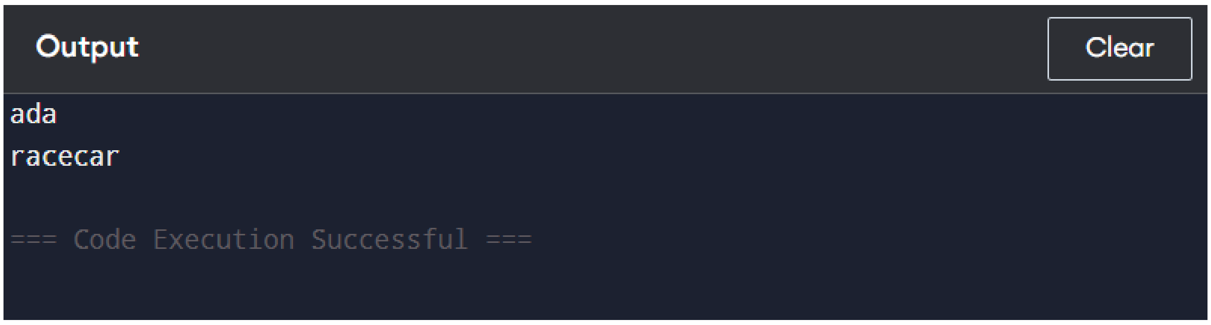
EXP 1: To Find and return the First palindromic string in a given list of words

CODE



```python
def firstPalindrome(words):
    for word in words:
        if word == word[::-1]:
            return word
    return ""
words1 = ["abc", "car", "ada", "racecar", "cool"]
print(firstPalindrome(words1))
words2 = ["notapalindrome", "racecar"]
print(firstPalindrome(words2))
```

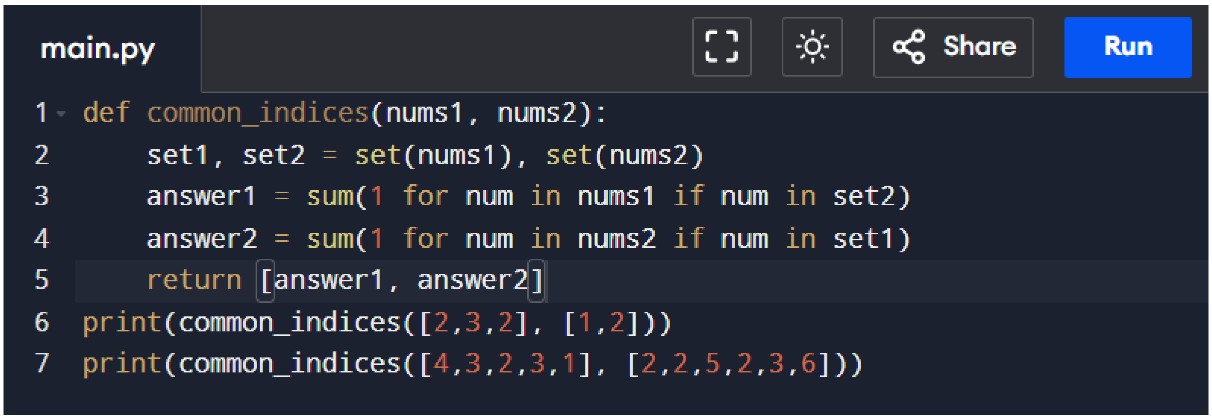
OUTPUT



```
ada
racecar

=== Code Execution Successful ===
```

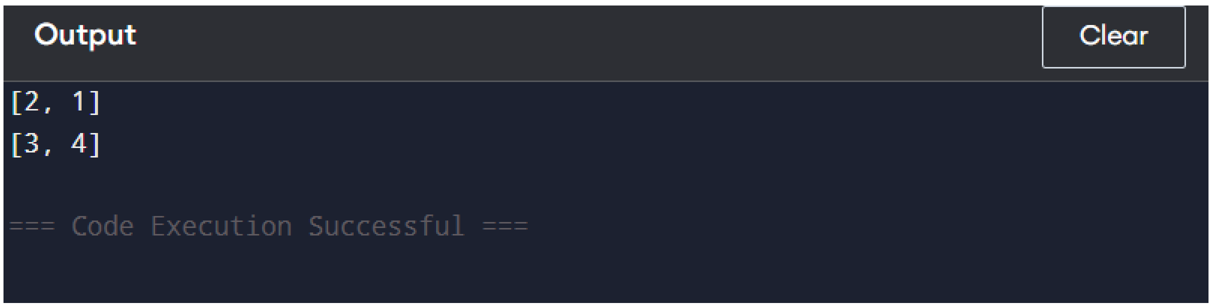
EXP 2: To find two integer arrays Nums1 and Nums2 of sizes n and m

CODE



```python
def common_indices(nums1, nums2):
    set1, set2 = set(nums1), set(nums2)
    answer1 = sum(1 for num in nums1 if num in set2)
    answer2 = sum(1 for num in nums2 if num in set1)
    return [answer1, answer2]
print(common_indices([2,3,2], [1,2]))
print(common_indices([4,3,2,3,1], [2,2,5,2,3,6]))
```

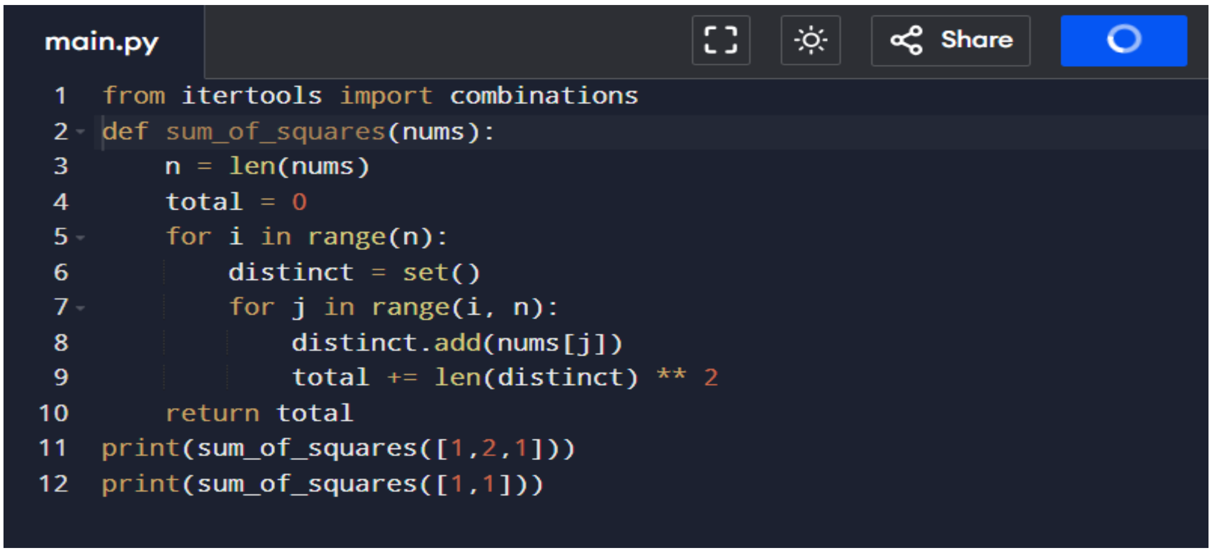
OUTPUT



```
[2, 1]
[3, 4]

=== Code Execution Successful ===
```

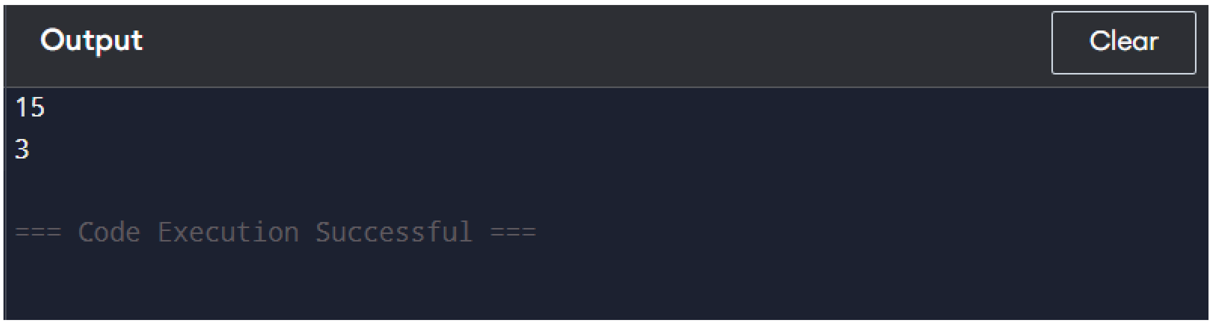
EXP 3 : To find the Sum of the Squares of distinct counts of all subarrays of a given list of integers

CODE



```python
from itertools import combinations
def sum_of_squares(nums):
    n = len(nums)
    total = 0
    for i in range(n):
        distinct = set()
        for j in range(i, n):
            distinct.add(nums[j])
            total += len(distinct) ** 2
    return total
print(sum_of_squares([1,2,1]))
print(sum_of_squares([1,1]))
```

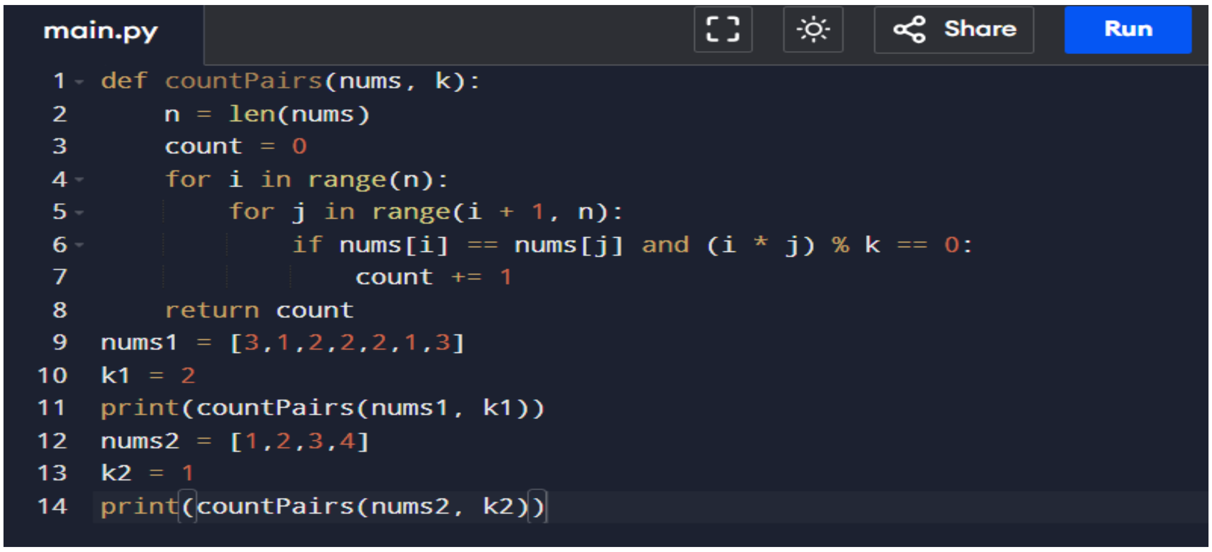
OUTPUT



```
15
3

=== Code Execution Successful ===
```

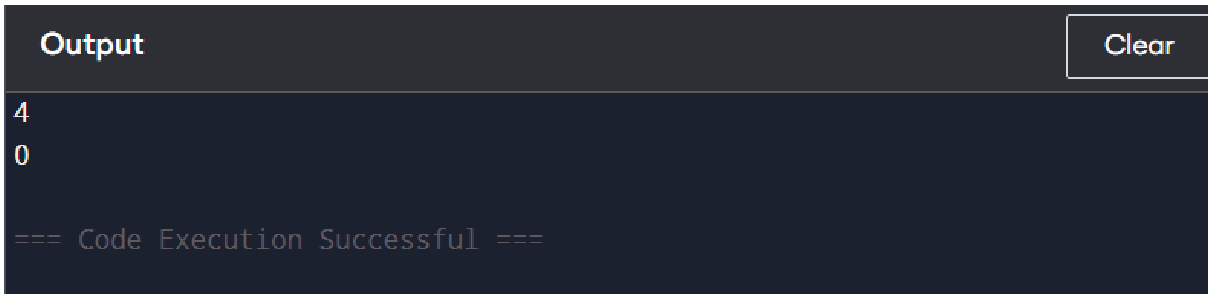
EXP 4 : Program to Count pairs in a array where elements are equal and the product of indices is divisible by a given number

CODE



```
def countPairs(nums, k):
    n = len(nums)
    count = 0
    for i in range(n):
        for j in range(i + 1, n):
            if nums[i] == nums[j] and (i * j) % k == 0:
                count += 1
    return count
nums1 = [3,1,2,2,2,1,3]
k1 = 2
print(countPairs(nums1, k1))
nums2 = [1,2,3,4]
k2 = 1
print(countPairs(nums2, k2))
```

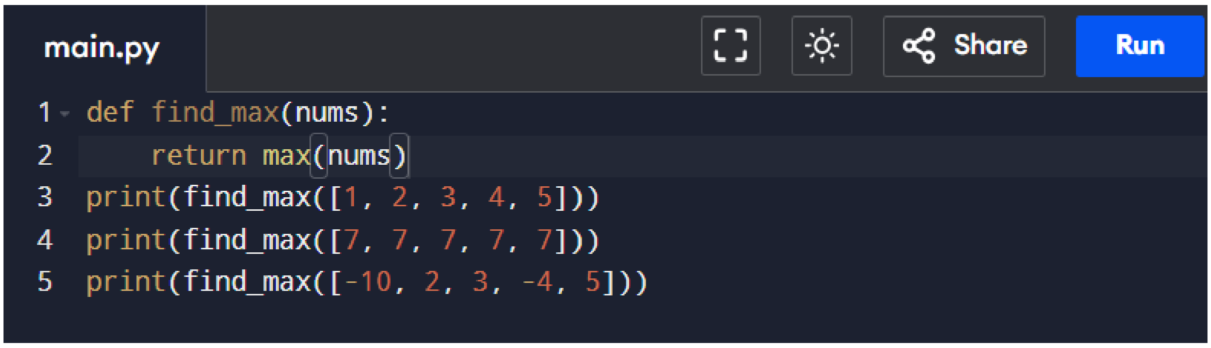
OUTPUT



```
4
0

=== Code Execution Successful ===
```

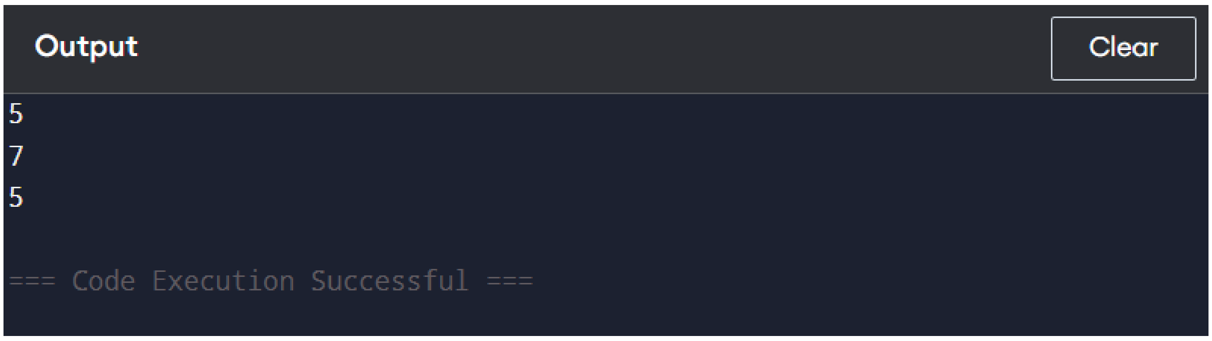
EXP 5 : Program to find the Maximum Element in an array

CODE



```python
def find_max(nums):
    return max(nums)
print(find_max([1, 2, 3, 4, 5]))
print(find_max([7, 7, 7, 7, 7]))
print(find_max([-10, 2, 3, -4, 5]))
```

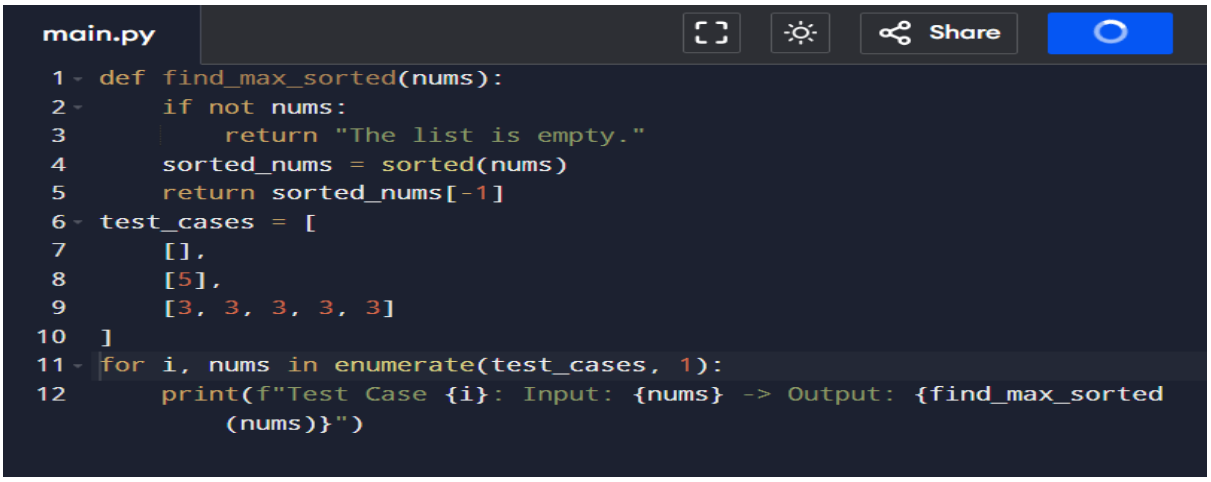
OUTPUT



```
5
7
5

=== Code Execution Successful ===
```

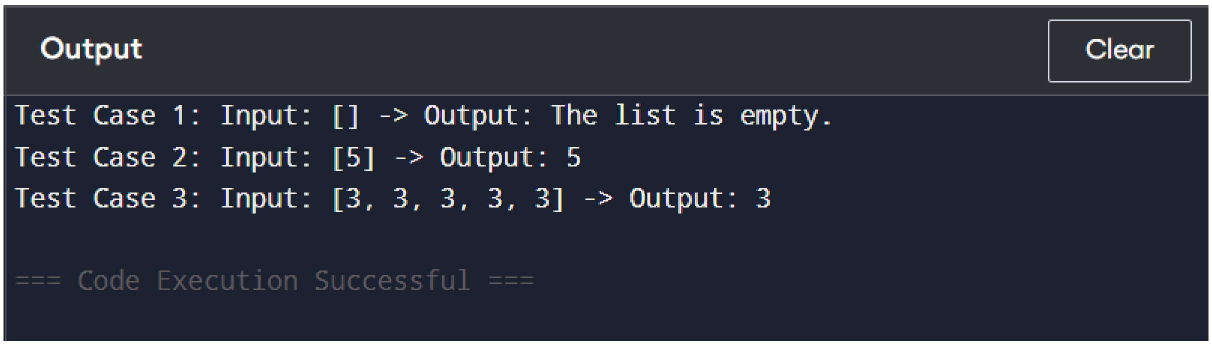
EXP 6 : Find Maximum Element in a list using sorting

CODE



```python
def find_max_sorted(nums):
    if not nums:
        return "The list is empty."
    sorted_nums = sorted(nums)
    return sorted_nums[-1]
test_cases = [
    [],
    [5],
    [3, 3, 3, 3, 3]
]
for i, nums in enumerate(test_cases, 1):
    print(f"Test Case {i}: Input: {nums} -> Output: {find_max_sorted
        (nums)}")
```

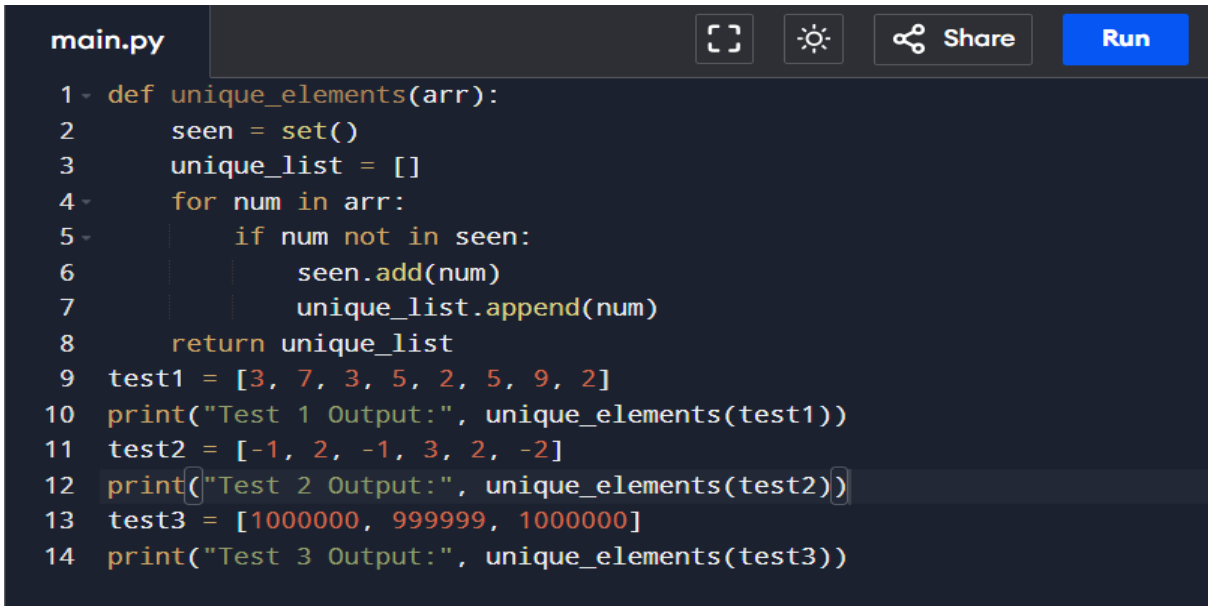
OUTPUT



```
Test Case 1: Input: [] -> Output: The list is empty.
Test Case 2: Input: [5] -> Output: 5
Test Case 3: Input: [3, 3, 3, 3, 3] -> Output: 3

=== Code Execution Successful ===
```

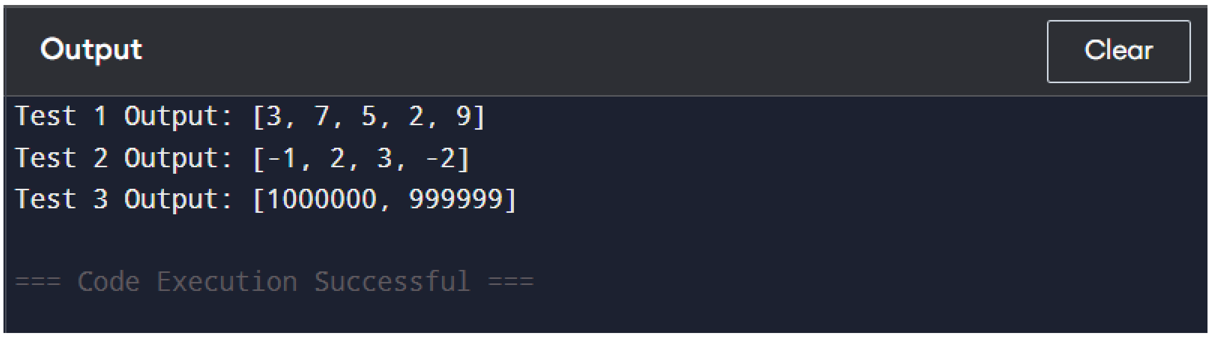
EXP 7 : Extract Unique element from a list

CODE



```python
def unique_elements(arr):
    seen = set()
    unique_list = []
    for num in arr:
        if num not in seen:
            seen.add(num)
            unique_list.append(num)
    return unique_list
test1 = [3, 7, 3, 5, 2, 5, 9, 2]
print("Test 1 Output:", unique_elements(test1))
test2 = [-1, 2, -1, 3, 2, -2]
print("Test 2 Output:", unique_elements(test2))
test3 = [1000000, 999999, 1000000]
print("Test 3 Output:", unique_elements(test3))
```

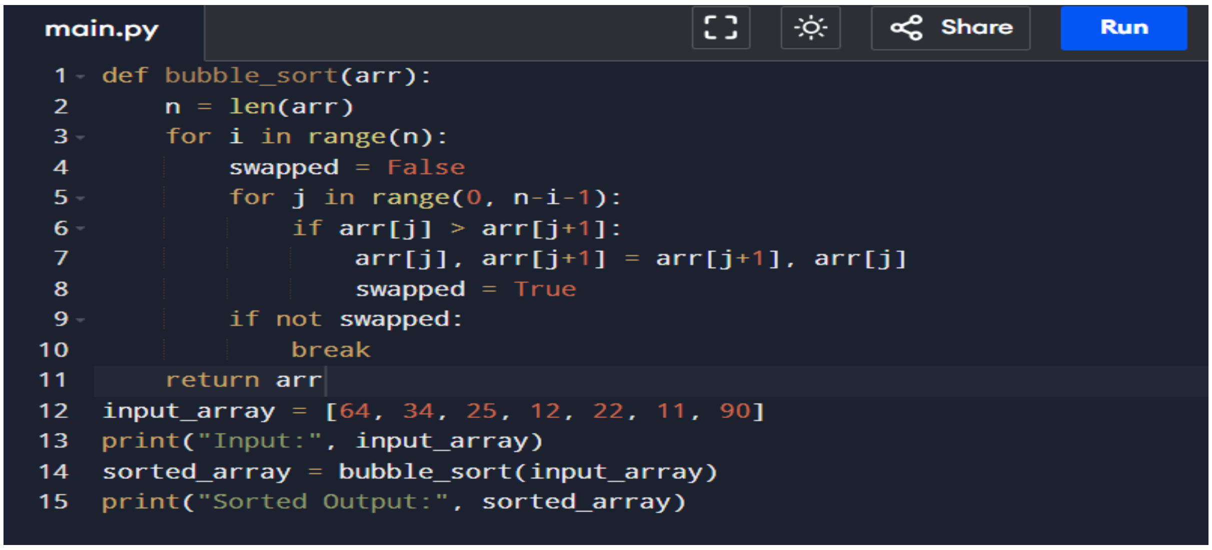
OUTPUT



```
Test 1 Output: [3, 7, 5, 2, 9]
Test 2 Output: [-1, 2, 3, -2]
Test 3 Output: [1000000, 999999]


=== Code Execution Successful ===
```

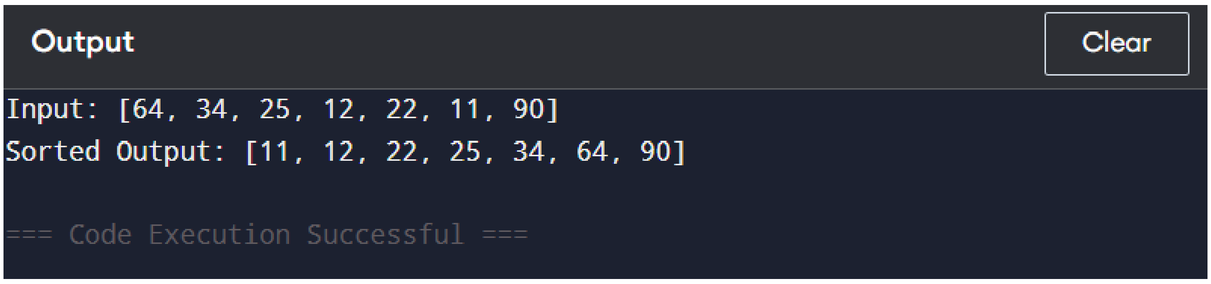
EXP 8: Bubble sort algorithm

CODE



```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        if not swapped:
            break
    return arr
input_array = [64, 34, 25, 12, 22, 11, 90]
print("Input:", input_array)
sorted_array = bubble_sort(input_array)
print("Sorted Output:", sorted_array)
```

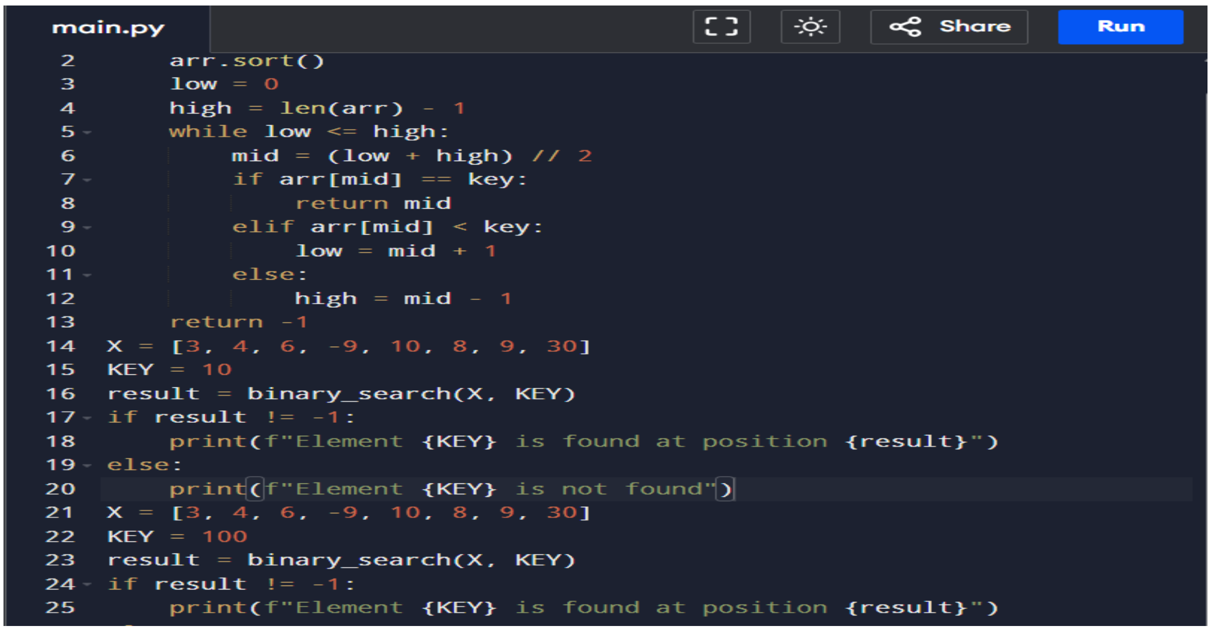
OUTPUT



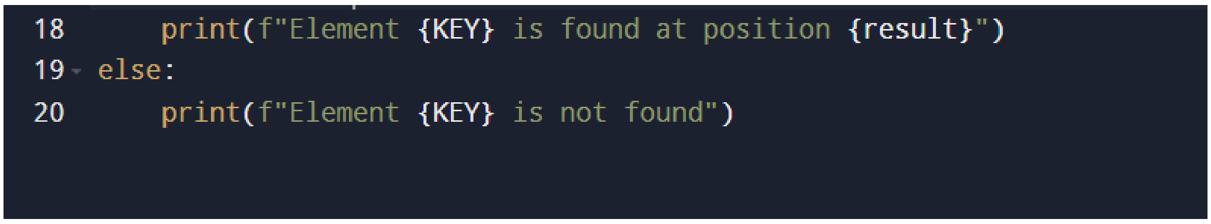
```
Input: [64, 34, 25, 12, 22, 11, 90]
Sorted Output: [11, 12, 22, 25, 34, 64, 90]

=== Code Execution Successful ===
```

EXP 9: Binary Search to check element existence in a sorted array

CODE

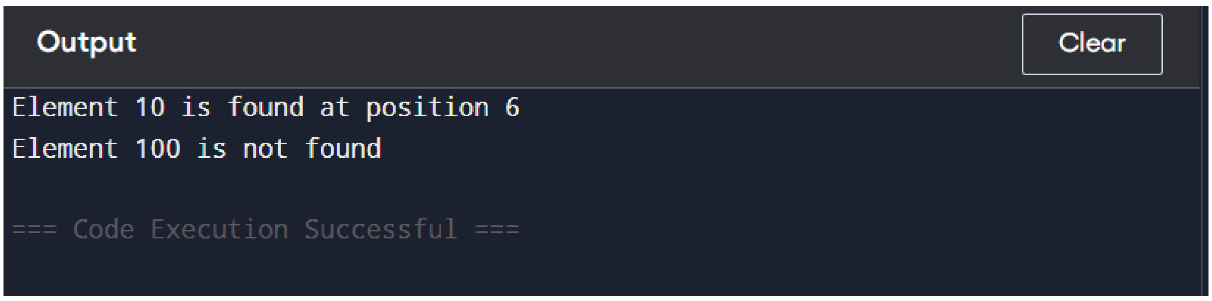


```python
    arr.sort()
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == key:
            return mid
        elif arr[mid] < key:
            low = mid + 1
        else:
            high = mid - 1
    return -1
X = [3, 4, 6, -9, 10, 8, 9, 30]
KEY = 10
result = binary_search(X, KEY)
if result != -1:
    print(f"Element {KEY} is found at position {result}")
else:
    print(f"Element {KEY} is not found")
X = [3, 4, 6, -9, 10, 8, 9, 30]
KEY = 100
result = binary_search(X, KEY)
if result != -1:
    print(f"Element {KEY} is found at position {result}")
```



```python
    print(f"Element {KEY} is found at position {result}")
else:
    print(f"Element {KEY} is not found")
```

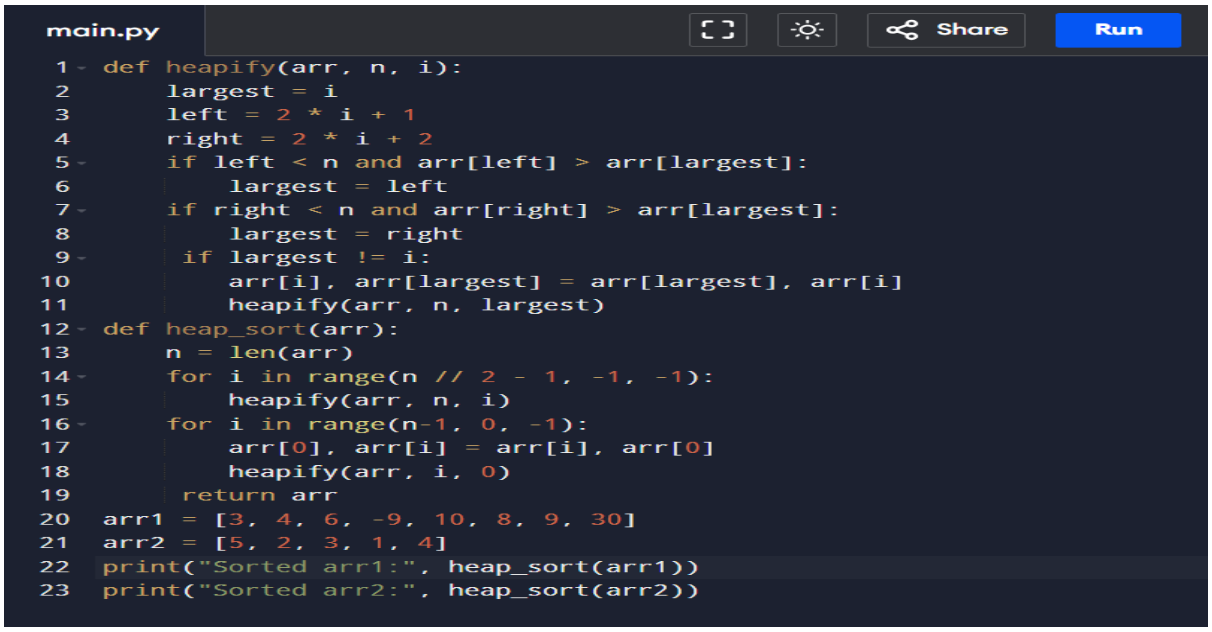
OUTPUT



```
Element 10 is found at position 6
Element 100 is not found


=== Code Execution Successful ===
```

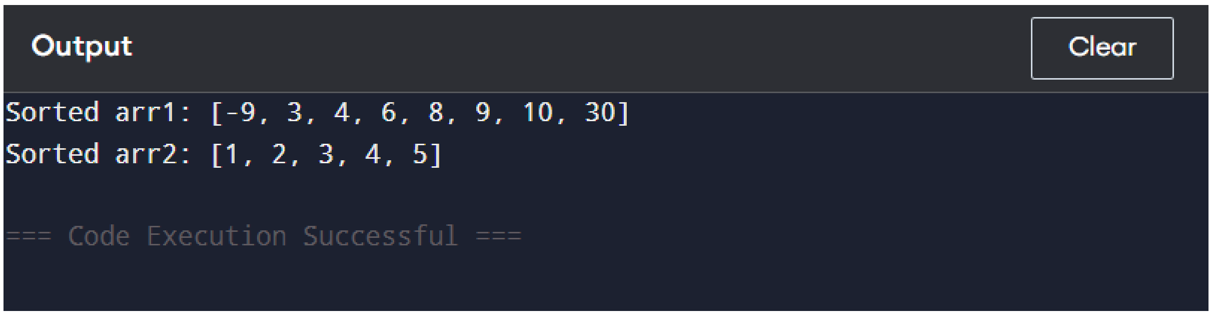
EXP 10 : Sort Array in ascending order using heap sort

CODE



```python
def heapify(arr, n, i):
    largest = i
    left = 2 * i + 1
    right = 2 * i + 2
    if left < n and arr[left] > arr[largest]:
        largest = left
    if right < n and arr[right] > arr[largest]:
        largest = right
     if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify(arr, n, largest)
def heap_sort(arr):
    n = len(arr)
    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)
    for i in range(n-1, 0, -1):
        arr[0], arr[i] = arr[i], arr[0]
        heapify(arr, i, 0)
     return arr
arr1 = [3, 4, 6, -9, 10, 8, 9, 30]
arr2 = [5, 2, 3, 1, 4]
print("Sorted arr1:", heap_sort(arr1))
print("Sorted arr2:", heap_sort(arr2))
```

OUTPUT



```
Sorted arr1: [-9, 3, 4, 6, 8, 9, 10, 30]
Sorted arr2: [1, 2, 3, 4, 5]


=== Code Execution Successful ===
```