67) **Given a collection of numbers, `nums`, that might contain duplicates, return *all possible unique permutations in any order.***

CODE:

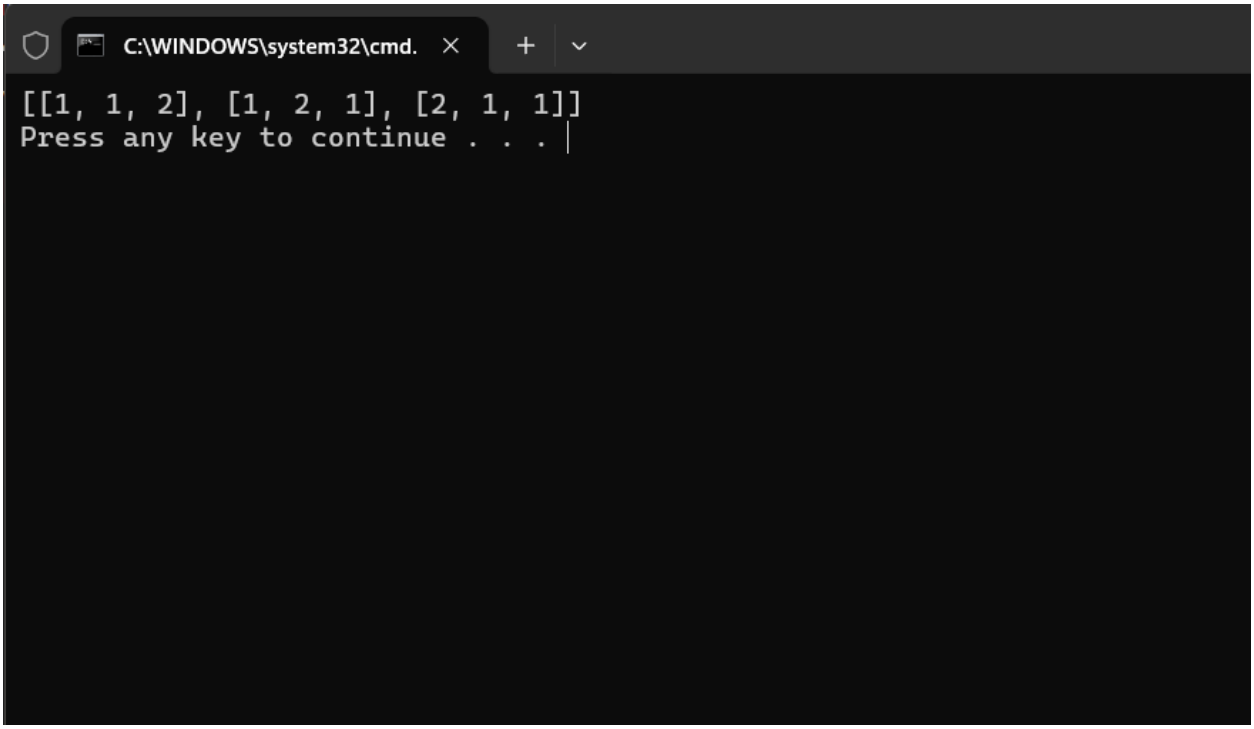
```
def permuteUnique(nums):
    nums.sort()
    result = []
    visited = [False] * len(nums)

    def backtrack(current_permutation):
        if len(current_permutation) == len(nums):
            result.append(list(current_permutation))
            return

        for i in range(len(nums)):
            if visited[i] or (i > 0 and nums[i] == nums[i - 1] and not visited[i -
1]):
                continue
            visited[i] = True
            current_permutation.append(nums[i])
            backtrack(current_permutation)
            current_permutation.pop()
            visited[i] = False

    backtrack([])
    return result
a=[1,1,2]
print(permuteUnique(a))
```

OUTPUT:



TIME COMPLEXITY :O(nlogn)