1.Merge 1wo sorted list

```
l1=[1,2,4]
l2=[1,3,4]
merge=l1+l2
sorted_list=sorted(merge)
print(sorted_list)
```

2.Merge k sorted list

```
l1=[1,4,4]
l2=[1,3,4]
l3=[2,6]
merge=l1+l2+l3
sorted_list=sorted(merge)
print(sorted_list)
```

3.Remove duplicates from sorted array

```
nums=[1,1,2]
list=list(set(nums))
print(len(list))
```

4.Search in rotated sorted array

```
def index(list,number):
    return list.index(number)
number=[4,5,6,7,0,1,2]
target=5
result=index(number,target)
print(result)
```

5.First and last position

```
nums = [5, 7, 7, 8, 8, 10]
target = 8
```

```
start = -1
end = -1
for i in range(len(nums)):
    if nums[i] == target:
        if start == -1:
            start = i
        end = i
result = [start, end]
print(result)
```

6.Sorr colors

```
nums=[2,0,2,1,1,0]
list=sorted(nums)
print(list)
```

7.Remove duplicates fron sorted list

```
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list
duplicate = [1,1,2]
print(Remove(duplicate))
```

8.Merge sorted array

```
l1=[1,2,3,0,0,0]
l2=[2,5,6]
merge=l1+l2
sorted_list=sorted(merge)
print(sorted_list)
```

9.Convert sorted array to bst

```
class TreeNode:
    def _init_(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def sortedArrayToBST(nums):
    if not nums:
        return None
    mid = len(nums) // 2
    root = TreeNode(nums[mid])
    root.left = sortedArrayToBST(nums[:mid])
    root.right = sortedArrayToBST(nums[mid + 1:])
    return root
nums1 = [-10, -3, 0, 5, 9]
result1 = sortedArrayToBST(nums1)
```

11.sort ferquency

```
def frequency_sort(s):
    char_freq = {}
    for char in s:
        char_freq[char] = char_freq.get(char, 0) + 1
    sorted_chars = sorted(s, key=lambda x: (-char_freq[x], x))
    return ''.join(sorted_chars)
input_str = "tree"
output_str = frequency_sort(input_str)
print(output_str)
```

12.max chunks are sorted

```
def max_chunks_to_sorted(arr):
    chunks = 0
```

```
    max_val = 0
    for i, val in enumerate(arr):
        max_val = max(max_val, val)
        if max_val == i:
            chunks += 1
    return chunks
arr = [4, 3, 2, 1, 0]
print(max_chunks_to_sorted(arr))
```

13. Intersection

```
def arrays_intersection(arr1, arr2, arr3):
    return sorted(set(arr1) & set(arr2) & set(arr3))
arr1 = [1, 2, 3, 4, 5]
arr2 = [1, 2, 5, 7, 9]
arr3 = [1, 3, 4, 5, 8]
print(arrays_intersection(arr1, arr2, arr3))
```

14.Sort the matrix diogonally

```
def diagonalSort(mat):
    m, n = len(mat), len(mat[0])
    diagonals = collections.defaultdict(list)
    for i in range(m):
        for j in range(n):
            diagonals[i - j].append(mat[i][j])
    for k in diagonals:
        diagonals[k].sort(reverse=True)
    for i in range(m):
        for j in range(n):
            mat[i][j] = diagonals[i - j].pop()
    return mat
mat = [[3,3,1,1],[2,2,1,2],[1,1,1,2]]
```

```
print(diagonalSort(mat))
```