1. Merge two sorted lists

```
def m(a, b):
   h = []
   i, j = 0, 0
   while i < len(a) and j < len(b):
      if a[i] < b[j]:
         h.append(a[i])
         i += 1
      else:
         h.append(b[j])
         j += 1
   while i < len(a):
      h.append(a[i])
      i += 1
   while j < len(b):
      h.append(b[j])
      j += 1
   return h
a = [1,2,4]
b = [1,3,4]
print(m(a,b))
```

2. Merge k sorted list

```
def m(l):
   def mt(a, b):
      h = []
      i, j = 0, 0
      while i < len(a) and j < len(b):
         if a[i] < b[j]:
            h.append(a[i])
            i += 1
         else:
            h.append(b[j])
            j += 1
        while i < len(a):
            h.append(a[i])
            i += 1
        while j < len(b):
            h.append(b[j])
            j += 1
        return h
    while len(l) > 1:
        ml = []
        for i in range(0, len(l), 2):
            l1 = l[i]
            l2 = l[i + 1] if (i + 1) < len(l) else []
            ml.append(mt(l1, l2))
        l = ml
    return l[0]
l = [[1,3,2,5], [5,26,9], [8,0,7]]
print(sorted(m(l)))
```

3. Remove duplicates from sorted array

```
def rem(a):
    if not a:
        return 0
    d = []
    for i in a:
        if i not in d:
            d.append(i)
    return d
a = [1,2,2,3,6,7,8,8,9]
print(rem(a))
```

4. Search in rotated sorted array

```
def s(a, t):
  l, h = 0, len(a) - 1
  while l <= h:
    m = (l + h) // 2
    if a[m] == t:
      return m
    if a[l] <= a[m]:
      if a[l] <= t < a[m]:
        h = m - 1
      else:
        l = m + 1
    else:
      if a[m] < t <= a[h]:
        l = m + 1
      else:
        h = m - 1
  return -1
a = [4,5,6,7,0,1,2]
t = 0
print(s(a,t))
```

5. Find First and Last Position of Element in Sorted Array

```
def fl(a,t):
  c=[]
  for i in range(len(a)):
    if a[i]==t:
      c.append(i)
    else:
      continue
  if not c:
        return [-1,-1]
    elif len(c)==1:
        return [c[0],-1]
    return c
a=[1,2,2,4,5,8,8]
t=9
print(fl(a,t))
```

6. Sort Colors

```
def col(ar):
    if len(ar)>1:
        mid = len(ar)//2
        l=col(ar[:mid])
        r=col(ar[mid:])
        i=j=k=0
        a=[0]*len(ar)
        while i<len(l) and j<len(r):
            if l[i]<r[j]:
                a[k]=l[i]
                i+=1
            else:
                a[k]=r[j]
                j+=1
            k+=1
        while i<len(l):
            a[k]=l[i]
            i+=1
            k+=1
        while j<len(r):
            a[k]=r[j]
            j+=1
            k+=1
        return a
    else:
        return ar
ar=[0,2,1,2,0,2,0,1]
print(col(ar))
```

7. Remove Duplicates from Sorted List

```
class L:
    def _init_(s, v=0, n=None):
        s.v = v
        s.n = n

def r(h):
    c = h
    while c and c.n:
        if c.v == c.n.v:
            c.n = c.n.n
        else:
            c = c.n
    return h

def p(node):
    while node:
        print(node.v, end=" ")
        node = node.n

h2 = L(1, L(1, L(2, L(3, L(3)))))
x =r(h2)
p(x)
```

8. Merge Sorted Array

```
def m(a1, m, a2, n):
    i, j, k = m-1, n-1, m+n-1
    while i >= 0 and j >= 0:
        if a1[i] > a2[j]:
            a1[k] = a1[i]
            i -= 1
        else:
            a1[k] = a2[j]
            j -= 1
        k -= 1
    while j >= 0:
        a1[k] = a2[j]
        j -= 1
        k -= 1
a1 = [1,2,3,0,0,0]
m1 = 3
a2 = [2,5,6]
n1 = 3
m(a1, m1, a2, n1)
print(a1)
```

9. Convert Sorted Array to Binary Search Tree

```
class T:
    def _init_(self, v=0, l=None, r=None):
        self.v = v
        self.l = l
        self.r = r


def c(a):
    if not a:
        return None
    m = len(a) // 2
    r = T(a[m])
    r.l = c(a[:m])
    r.r = c(a[m+1:])
    return r

def print_tree(node):
    if node:
        print(node.v, end=" ")
        print_tree(node.l)
        print_tree(node.r)

a1 = [-10, -3, 0, 5, 9]
t1 = c(a1)
print_tree(t1)
```

10. Insertion Sort List

```
class L:
    def _init_(self, v=0, n=None):
        self.v = v
        self.n = n

def i(h):
    if not h or not h.n:
        return h
    d = L(0)
    c = h
    while c:
        t = c.n
        p = d
        while p.n and p.n.v < c.v:
            p = p.n
        c.n = p.n
        p.n = c
        c = t
    return d.n

def print_list(node):
    while node:
        print(node.v, end=" ")
        node = node.n
    print()

h1 = L(4, L(2, L(1, L(3))))
s1 = i(h1)
print_list(s1)
```

11. Sort Characters By Frequency

```
from collections import Counter

def s(f):
    c = Counter(f)
    return ''.join([k * v for k, v in c.most_common()])

s1 = "tree"
o1 = s(s1)
print(o1)
```

12. Max Chunks To Make Sorted

```
def m(a):
    mx, c = 0, 0
    for i, n in enumerate(a):
        mx = max(mx, n)
        if mx == i:
            c += 1
    return c

a1 = [4, 3, 2, 1, 0]
o1 = m(a1)
print(o1)
```

13. Intersection of Three Sorted Arrays

```
def i(a1, a2, a3):
    s1, s2, s3 = set(a1), set(a2), set(a3)
    return sorted(s1 & s2 & s3)

a1_1 = [1, 2, 3, 4, 5]
a2_1 = [1, 2, 5, 7, 9]
a3_1 = [1, 3, 4, 5, 8]
o1 = i(a1_1, a2_1, a3_1)
print(o1)
```

14. Sort the Matrix Diagonally

```
from collections import defaultdict
import heapq

def s(m):
    d = defaultdict(list)
    for i in range(len(m)):
        for j in range(len(m[0])):
            heapq.heappush(d[i-j], m[i][j])
    for i in range(len(m)):
        for j in range(len(m[0])):
            m[i][j] = heapq.heappop(d[i-j])
    return m



m1 = [[3, 3, 1, 1],[2, 2, 1, 2],[1, 1, 1, 2]]
o1 = s(m1)
print(o1)
```