```
def rightRotation(word1, word2):
      n = len(word1)
      m = len(word2)
      if word1 == None or word2 == None or n == 0 or m == 0 or n != m:
            return -1
      ans = word1 + word2
      return 1 if ans.index(word2) != -1 else -1


def longestPalindrome(s):
      n = len(s)
      ans = ""
      nums = 0
      if n == 0:
            return ans
      for i in range(n*2-1):
            left = i//2
            right = (i+1)//2
            while left >= 0 and right < n and s[left] == s[right]:
                  left -=1
                  right +=1
            if nums < right-left+1:
                  nums =  right-left+1
                  ans = s[left+1:right]
      return ans


def mergeTwoLists(l1,l2):
      if not l1 or not l2:
            return l1 if not l2 else l2

      if l2:
            if l1.val > l2.val:
                  l2,l1 = l1,l2
            l1.next = self.mergeTwoLists(l1.next, l2)
      return l1


# 这里是另一种解法 可以把 tree 变成 string 然后直接看 s 是不是包含了 t
 def isSubtree(s,t):
      def convert(p):
            return "^" + str(p.val) + "#" + convert(p.left) + convert(p.right)
if p else "$"
      return convert(t) in convert(s)

def highFive(items):
      d = collections.defaultdict(list)
      for idx, val in items:
            heapq.heappush(d[idx], val)
            if len(d[idx]) > 5:
                  heapq.heappop(d[idx])
      res = [[i, sum(d[i])//5] for i in sorted(d)]
      return res
```