```
# -*- coding: utf-8 -*-
#Applied algo Question 3 - Combined Signals Strength
def combine_signals(a):
    cmb_str=[]
    x=0
    while(x!=(len(a)-1)):
        k=a[x]|a[x+1]
        cmb_str.append(k)
        x+=1
    return cmb_str
#signals=[int(x) for x in input().split()]
#print(combine_signals(signals))

#Applied algo Question 4 - Elite teams
def numberOfTeams(ratings)->int:
    lnt= len(ratings)
    ll = [0]*lnt
    rl = [0]*lnt
    ls = [0]*lnt
    rs = [0]*lnt
    for i in range(lnt):
        for j in range(i):
            if ratings[j]<ratings[i]:
                ls[i]=ls[i]+1
            elif ratings[j]>ratings[i]:
                ll[i]=ll[i]+ 1

    for i in range(lnt):
        for j in range(i + 1,lnt):
            if ratings[j]>ratings[i]:
                rl[i]=rl[i]+1
            elif ratings[j]<ratings[i]:
                rs[i]=rs[i]+1

    Teamssum=0
    for i in range(lnt):
        Teamssum=Teamssum+ls[i]*rl[i]
        Teamssum+=ll[i]*rs[i]
    return Teamssum

#ratings = [2,1,3]
#print(numberOfTeams(ratings))  # Output: 3

#Question 5 - Pirate Crew Task
def findPowerLevels(pwrLvls) -> list[int]:
    n=len(pwrLvls)
    nArr=[1 for i in range(n)]
    lp=1
    rp=1
    x=0
    while(x!=n):
        nArr[x]=lp
        lp=lp*pwrLvls[x]
        x+=1

    for j in range(n-1,-1,-1):
        nArr[j]=nArr[j]*rp
        rp=rp*pwrLvls[j]
    return nArr
```

```
#powerLevels = [1, 2, 3, 4]
#print(findPowerLevels(powerLevels))

#Question 6 - Anagram Half Match

from collections import Counter
def decrypt(s:str)->int:
    stl=len(s)
    if stl%2!=0:
        return -1
    mid=stl//2
    fh=s[:mid]
    sh=s[mid:]
    fc=Counter(fh)
    sc=Counter(sh)
    ch=0
    for i in fc:
        if fc[i]>sc[i]:
            ch=ch+fc[i]-sc[i]
    return ch

#s='xyyx'
#print(decrypt(s))

#Question 7
def magicalScribe(n:int) -> str:
    if n==1:
        return "1"
    def next_seq(seq):
        result=[]
        i = 0
        while i<len(seq):
            count=1
            while i+1<len(seq) and seq[i]==seq[i+1]:
                i=i+1
                count=count+1
            result.append(str(count)+seq[i])
            i=i+1
        return ''.join(result)
    current_seq = "1"
    for i in range(2,n+1):
        current_seq= next_seq(current_seq)
    return current_seq

#n = 2
#print(magicalScribe(n))

#Question 8 - Vowel Sort Decode
def sortVowels(code):
    v=('aeiouAEIOU')
    vwls_array=[x for x in code if x in v]
    vwls_array.sort()
    res=[]
    v_index=0
    for x in code:
        if x in v:
           res.append(vwls_array[v_index])
           v_index+=1
```

```
        else:
           res.append(x)
    result=''.join(res)
    return result

#message = "hEllo bAtmAn"
#print(sortVowels(message))

#Question 9 - Help Elara
def helpElara(words, width):
    res=[]
    line=[]
    line_len=0
    for word in words:
        if line_len+len(word)+len(line)>width:
            for i in range(width-line_len):
                line[i%(len(line)-1 or 1)]+= ' '
            res.append(''.join(line))
            line=[]
            line_len=0
        line.append(word)
        line_len+=len(word)
    res.append(' '.join(line).ljust(width))
    return res


#Question 10 - Cyclonia’s Quest
class ListNode:
    def __init__(self, x):
        self.val = x
        self.next = None

def findTreasure(head:ListNode):
    if not head:
      return True
    tort=head
    hare=head
    while (hare and hare.next):
        tort=tort.next
        hare=hare.next.next
        if tort==hare:
            return False
    return True

#Question 1
from typing import Union
from collections import OrderedDict
def processOperations(n:int,operations:list[tuple[int,Union[str,
tuple[str,int]]]]):
    mv_rtg={}
    res=[]
    acc_cnt=OrderedDict()
    for y in operations:
        one=y[0]
        two=y[1]
        if(one==1 and isinstance(two, str)):
            if(two in mv_rtg):
               res.append(mv_rtg.get(two))
               acc_cnt[two]+=1
```

```
            if(len(acc_cnt)>1):
                acc_cnt.move_to_end(two)
        else:
            res.append(-1)

    elif(one==2 and isinstance(two,tuple)):
        count=0
        mv_rtg[two[0]]=two[1]
        if(len(mv_rtg)>n):
            min_value=min(acc_cnt.values())
            for key in acc_cnt:
                if acc_cnt[key]==min_value:
                    min_value_key=key
                    break
            mv_rtg.pop(min_value_key)
            acc_cnt.pop(min_value_key)
        if(two[0]in acc_cnt):
            acc_cnt[two[0]]+=1
        else:
            count=count+1
            acc_cnt[two[0]]=count
        if(len(acc_cnt)>1):
               acc_cnt.move_to_end(two[0])
return res
```