ZAIN-UL-ABIDEEN

REG# 450682

Contents

[1.1 Hard exercise python heckerrank 2](#_Toc154800207)

[1.2 Medium exercise python heckerrank 2](#_Toc154800208)

[1.1.1 Maximize it! 2](#_Toc154800209)

[1.1.2 Validating Postal Codes. 3](#_Toc154800210)

[1.1.3 Matrix Script. 3](#_Toc154800211)

[1.2.1 Write a function. 4](#_Toc154800212)

[1.2.2 The Minion Game. 5](#_Toc154800213)

[1.2.3 Merge the Tools! 5](#_Toc154800214)

[1.2.4 Time Delta. 6](#_Toc154800215)

[1.2.5 Find the Angle MBC. 7](#_Toc154800216)

[1.2.6 No Idea! 7](#_Toc154800217)

[1.2.7 Word order. 8](#_Toc154800218)

[1.2.8 Compress the string. 8](#_Toc154800219)

[1.2.9 Company logo. 9](#_Toc154800220)

[1.2.10 Piling Up 9](#_Toc154800221)

[1.2.11 Triangle Quest 2. 10](#_Toc154800222)

[1.2.12 Iterables & iterators 10](#_Toc154800223)

[1.2.13 Triangle Quest 10](#_Toc154800224)

[1.2.14 Classes: Dealing with Complex Numbers. 11](#_Toc154800225)

[1.2.15 Athlete Sort. 12](#_Toc154800226)

[1.2.16 ginortS. 12](#_Toc154800227)

[1.2.17 Validating Email Addresses With a Filter. 13](#_Toc154800228)

[1.2.18 Reduce Function. 14](#_Toc154800229)

[1.2.19 Regex Substitution. 14](#_Toc154800230)

[15](#_Toc154800231)

[1.2.20 Validating Credit Card Numbers. 15](#_Toc154800232)

[1.2.21 Word Score. 15](#_Toc154800233)

[1.2.22 Default Arguments. 16](#_Toc154800234)

# Hard exercise python heckerrank

# Medium exercise python heckerrank

## Maximize it!

1. from itertools import product
2. k, m = list(map(int, input().split()))
3. #print(k, m)
4. lines = []
5. for item in range(k):
6. lines.append(list(map(int, input().split()))[1:])
8. #print(lines)
9. result\_list = list(product(\*lines))
10. #print(result\_list)
11. result = []
12. for tup in result\_list:
13. total = 0
14. for item in tup:
15. total = total + (item\*\*2)
17. result.append(total%m)
19. print(max(result))

## 1.1.2 Validating Postal Codes.

regex\_integer\_in\_range = r"^[1-9][\d]{5}$"  # Do not delete 'r'.

regex\_alternating\_repetitive\_digit\_pair = r"(\d)(?=\d\1)"   # Do not delete 'r'.

import re

P = input()

print (bool(re.match(regex\_integer\_in\_range, P))

and len(re.findall(regex\_alternating\_repetitive\_digit\_pair, P)) < 2)

## 1.1.3 Matrix Script.

#!/bin/python3

import math

import os

import random

import re

import sys

first\_multiple\_input = input().rstrip().split()

n = int(first\_multiple\_input[0])

m = int(first\_multiple\_input[1])

matrix = []

for \_ in range(n):

    matrix\_item = input()

    matrix.append(matrix\_item)

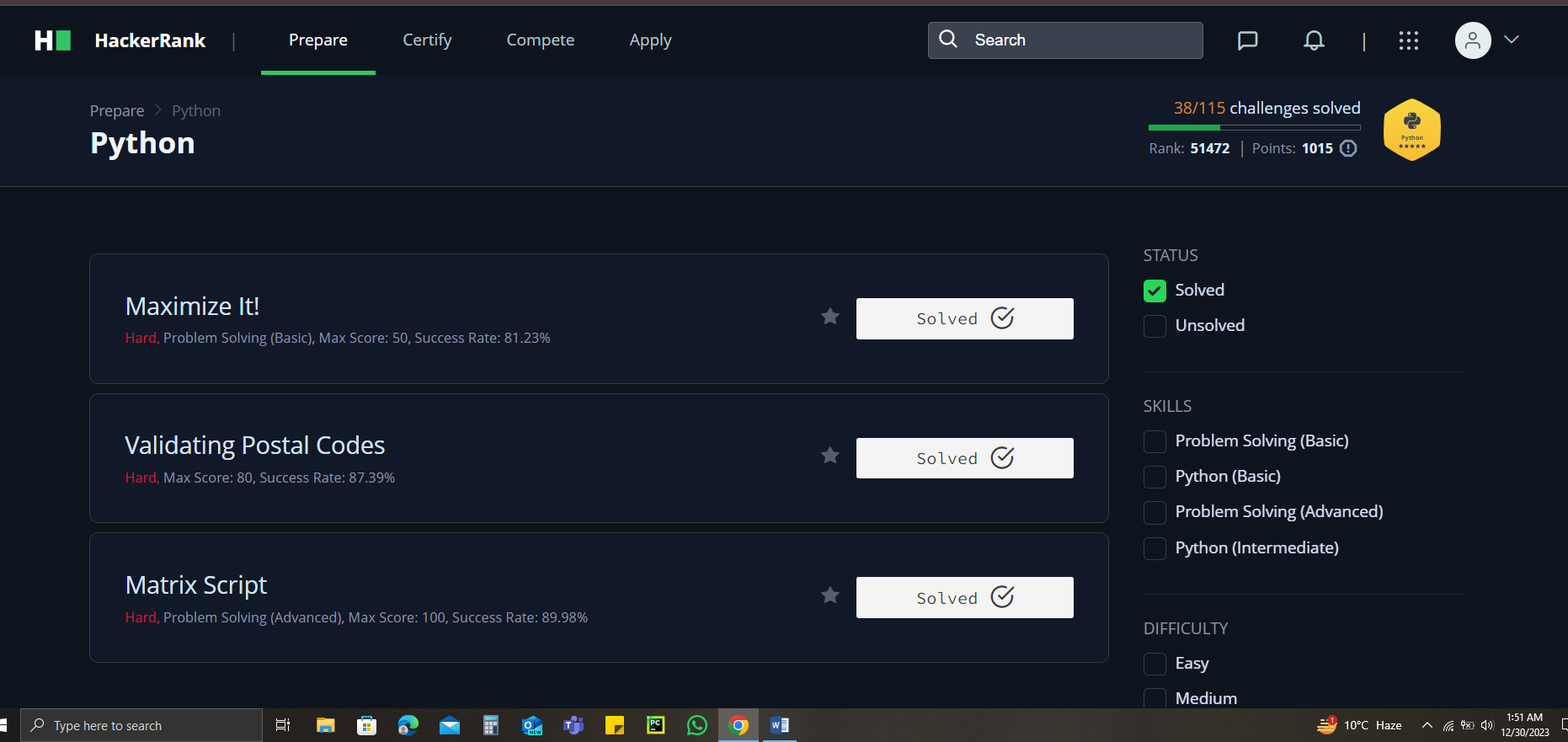
decoded\_list = list(zip(\*matrix))

decoded\_string = ''

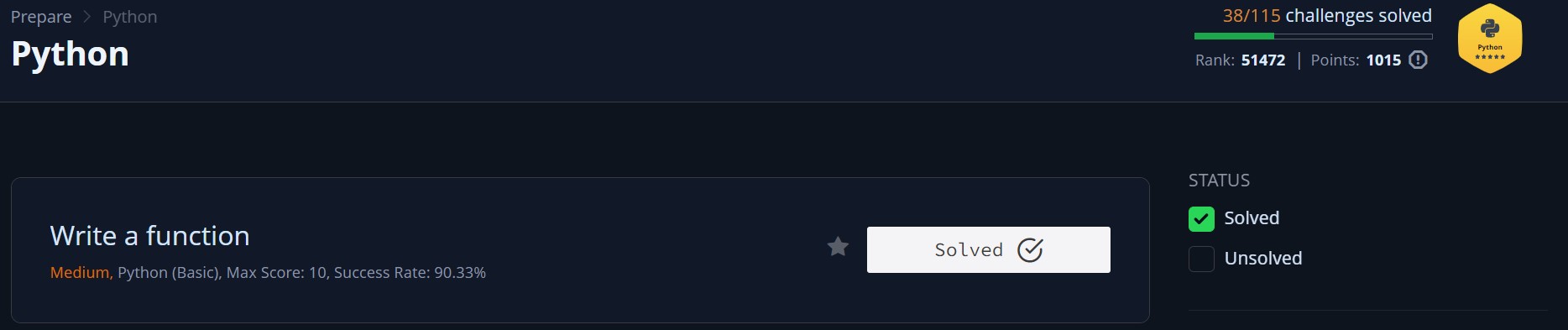
for item in decoded\_list:

    decoded\_string = decoded\_string + "".join(item)

print(re.sub(r'\b[^a-zA-Z0-9]+\b', r' ', decoded\_string))



## Write a function.

1. def is\_leap(year):
2. leap = False
4. if (year%4 ==0) and (year%100!=0 or year%400==0):
5. leap = True
7. return leap
8. year = int(input())
9. print(is\_leap(year))
10. 

## 1.2.2 The Minion Game.

def minion\_game(string):

   vowels = 'AEIOU'

   kevin\_score = stuart\_score =0

   length = len(string)

   for i in range(length):

       if string[i] in vowels:

           kevin\_score += length -i

        else:

            stuart\_score +=length -i

    if stuart\_score > kevin\_score:

        print(f'Stuart(stuart\_score)')

    elif kevin\_score > stuart\_score:

        print(f'Kevin(kevin\_score)')

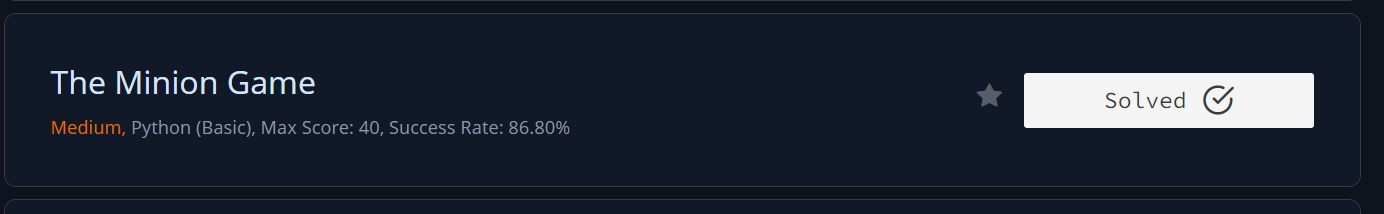
    else:

        print('Draw')

if \_\_name\_\_ == '\_\_main\_\_':

    s = input()

    minion\_game(s)



## 1.2.3 Merge the Tools!

def merge\_the\_tools(string, k):

    start = 0

    end = k

    while end<= len(string):

        temp = string[start:end]

        chunk = list(set(list(temp)))

        print(''.join(chunk))

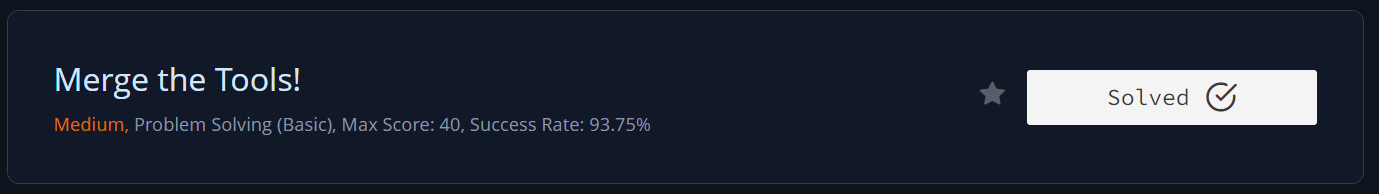
        start += k

        end += k

if \_\_name\_\_ == '\_\_main\_\_':

    string, k = input(), int(input())

    merge\_the\_tools(string, k)



## 1.2.4 Time Delta.

from datetime import datetime

def time\_delta(t1, t2):

    # Define the format of the timestamps

    fmt = '%a %d %b %Y %H:%M:%S %z'

    # Convert timestamps to datetime objects

    dt1 = datetime.strptime(t1, fmt)

    dt2 = datetime.strptime(t2, fmt)

    # Calculate the time difference

    delta = abs(dt1 - dt2)

    return int(delta.total\_seconds())

if \_\_name\_\_ == "\_\_main\_\_":

    # Read the number of test cases

    t = int(input().strip())

    for \_ in range(t):

        # Read two timestamps

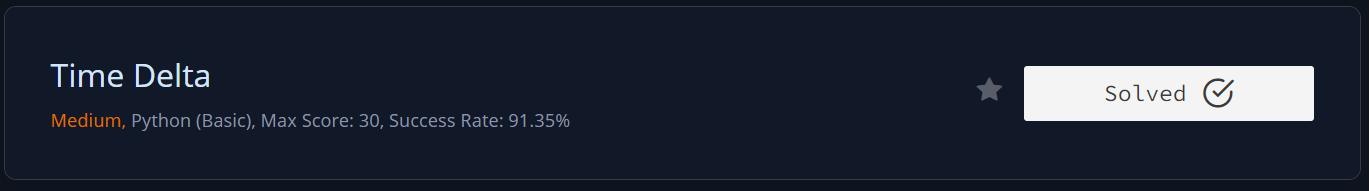
        timestamp1 = input().strip()

        timestamp2 = input().strip()

        # Calculate and print the time difference

        result = time\_delta(timestamp1, timestamp2)

        print(result)



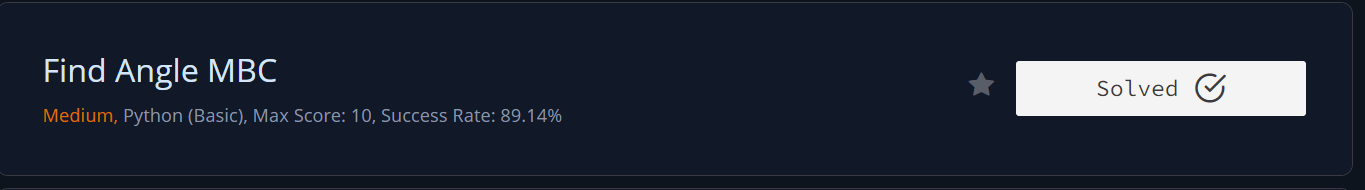
## 1.2.5 Find the Angle MBC.

import math

AB =int(input())

BC = int(input())

print(round(math.degrees(math.atan(AB/BC))),chr(176),sep='')



## 1.2.6 No Idea!

input()

n = input().split()

A = set(input().split())

B = set(input().split())

h = 0

for i in n:

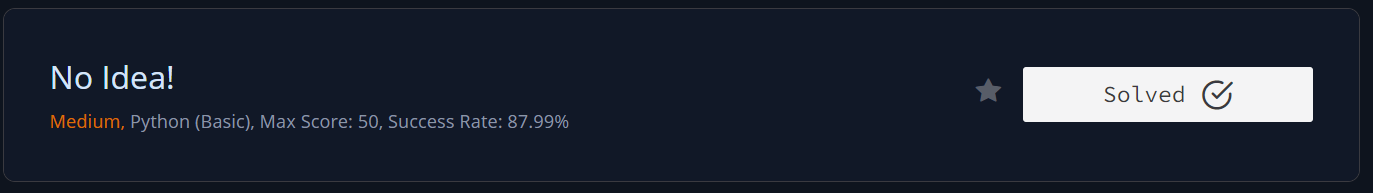
    if i in A:

        h+= 1

    elif i in B:

        h-= 1

print(h)



## 1.2.7 Word order.

from collections import OrderedDict

od = OrderedDict()

for i in range(int(input())):

    word = input()

    if word not in od:

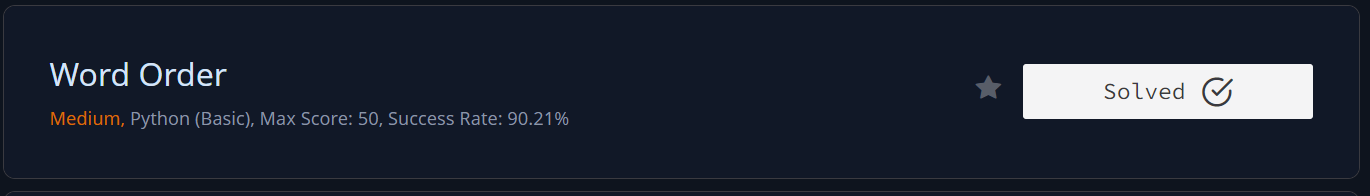
        od[word] = 1

    else:

        od[word] += 1

print(len(od))

print(\*od.values())



## 1.2.8 Compress the string.

s=input()

count = 1

for i in range(1, len(s)):

    if s[i]==s[i-1]:

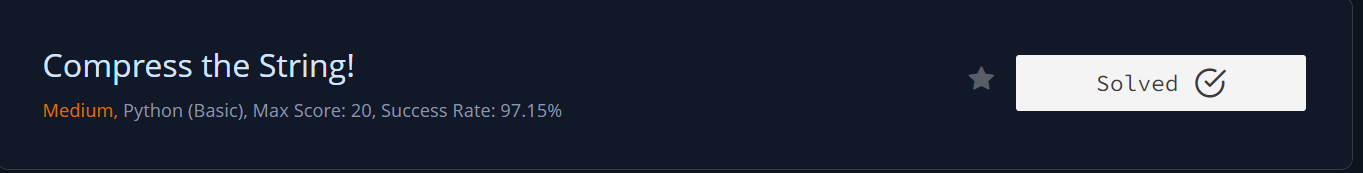
        count +=1

    else:

        print((count, int(s[i-1])), end=" ")

        count=1

print((count, int(s[-1])))



## 1.2.9 Company logo.

#!/bin/python3

import math

import os

import random

import re

import sys

from collections import Counter

if \_\_name\_\_ == '\_\_main\_\_':

    s = sorted(input())

    z= Counter(s)

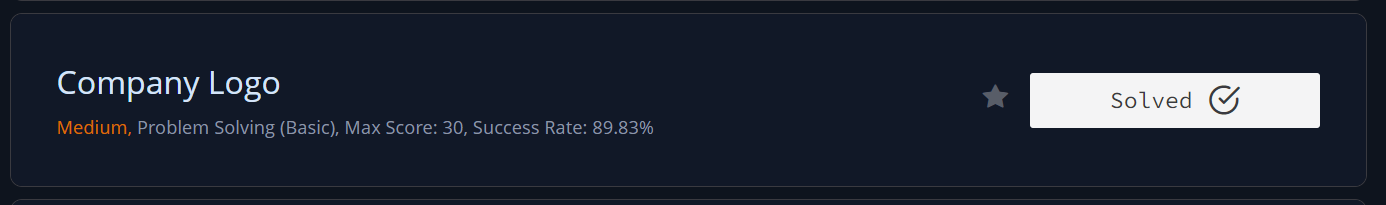
   # print(z)

    z=Counter(s).most\_common(3)

    # print(z)

    for x in z:

        print(\*x)



## 1.2.10 Piling Up

for t in range(int(input())):

    input()

    lst = [int(i) for i in input().split()]

    min\_list = lst.index(min(lst))

    left = lst[:min\_list]

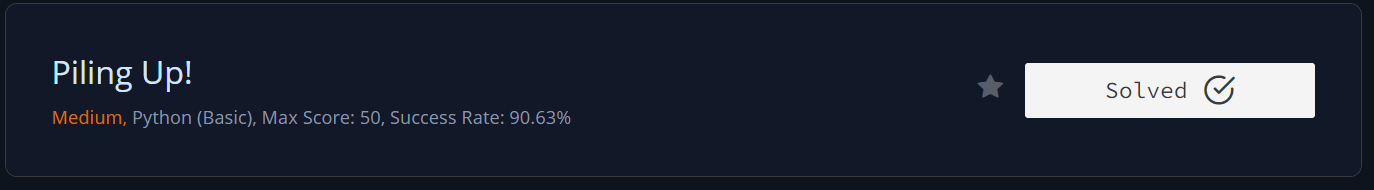
    right = lst[min\_list+1:]

    if left == sorted(left, reverse =True) and right == sorted(right):

        print("Yes")

    else:

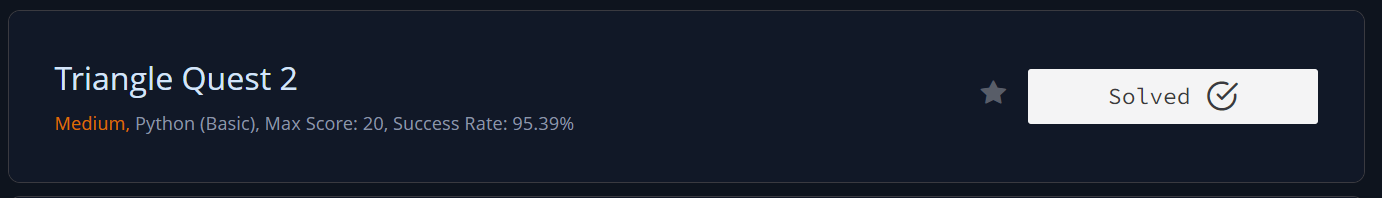
        print("No")



## 1.2.11 Triangle Quest 2.

for i in range(1,int(input())+1): #More than 2 lines will result in 0 score. Do not leave a blank line also

    print (((10\*\*i)//9)\*\*2)



## 1.2.12 Iterables & iterators

# Enter your code here. Read input from STDIN. Print output to STDOUT

from itertools import combinations

n = int(input())

l = input().split()

k = int(input())

l1 = list(combinations(l, k))

counta = 0

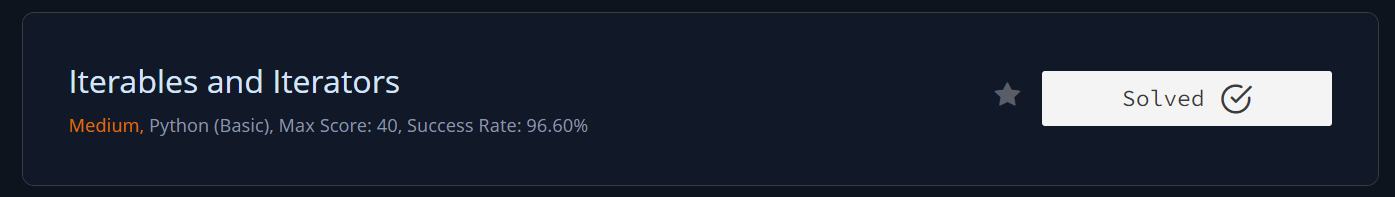
for i in l1:

    if "a" in i:

        counta +=1

p = counta / len(l1)

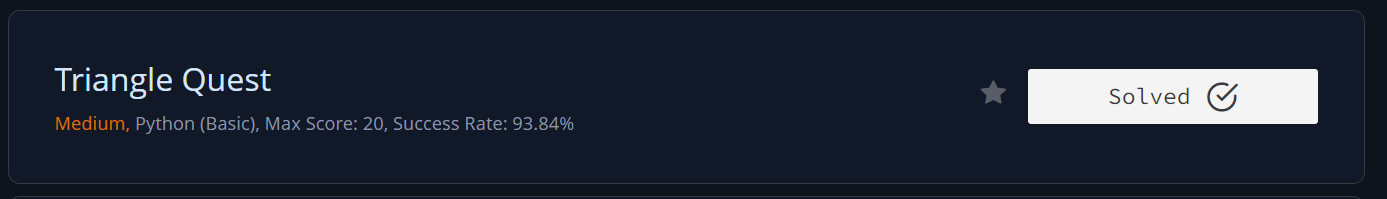
print("%.4f"%p)



## 1.2.13 Triangle Quest

for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank line also

    print(((10\*\*i)//9)\*i)



## 1.2.14 Classes: Dealing with Complex Numbers.

import math

class Complex(object):

    def \_\_init\_\_(self, real, imaginary):

        self.real = real

        self.imaginary = imaginary

    def \_\_add\_\_(self, no):

        return Complex(self.real+no.real, self.imaginary+no.imaginary)

    def \_\_sub\_\_(self, no):

        return Complex(self.real-no.real, self.imaginary-no.imaginary)

    def \_\_mul\_\_(self, no):

        r = self.real\*no.real - self.imaginary\*no.imaginary

        i = self.real\*no.imaginary + self.imaginary\*no.real

        return Complex(r, i)

    def \_\_truediv\_\_(self, no):

        d=no.real\*\*2+no.imaginary\*\*2

        n=self\*Complex(no.real, -1\*no.imaginary)

        return Complex(n.real/d, n.imaginary/d)

    def mod(self):

        d=self.real\*\*2+self.imaginary\*\*2

        return Complex(math.sqrt(d), 0)

    def \_\_str\_\_(self):

        if self.imaginary == 0:

            result = "%.2f+0.00i" % (self.real)

        elif self.real == 0:

            if self.imaginary >= 0:

                result = "0.00+%.2fi" % (self.imaginary)

            else:

                result = "0.00-%.2fi" % (abs(self.imaginary))

        elif self.imaginary > 0:

            result = "%.2f+%.2fi" % (self.real, self.imaginary)

        else:

            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))

        return result

if \_\_name\_\_ == '\_\_main\_\_':

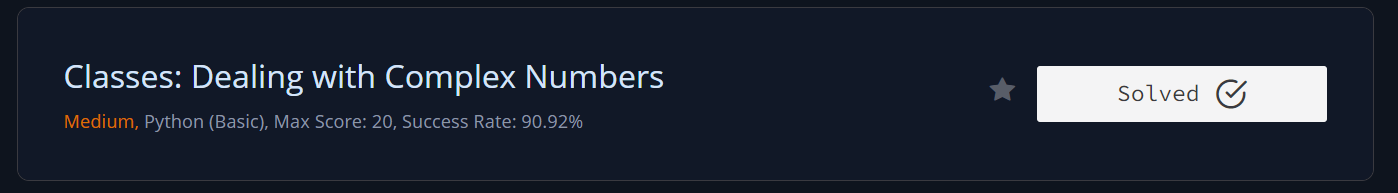
    c = map(float, input().split())

    d = map(float, input().split())

    x = Complex(\*c)

    y = Complex(\*d)

    print(\*map(str, [x+y, x-y, x\*y, x/y, x.mod(), y.mod()]), sep='\n')



## 1.2.15 Athlete Sort.

from operator import itemgetter

nm = list(map(int,input().split()))

n, m = nm[0], nm[1]

row = []

for i in range(n):

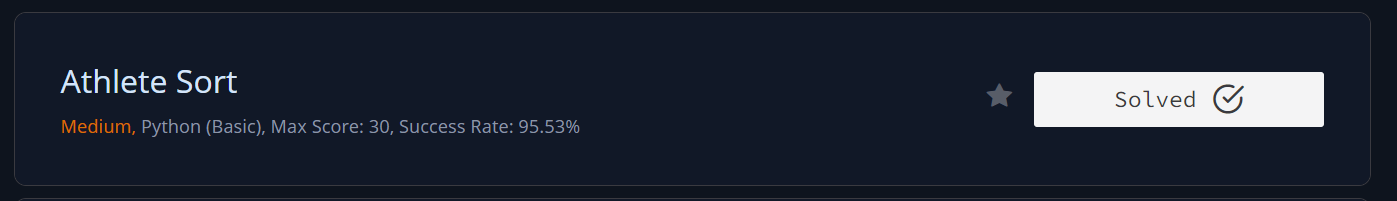
    row += [list(map(int,input().split()))]

k = int(input())

arr = sorted (row, key=itemgetter(k))

for i in arr:

    print(\*i)



## 1.2.16 ginortS.

# lower, upper, odd , even

lower = ""

upper = ""

even = ""

odd = ""

s= sorted(input())

for i in s:

    if i.islower():

        lower += i

    elif i.isupper():

        upper+= i

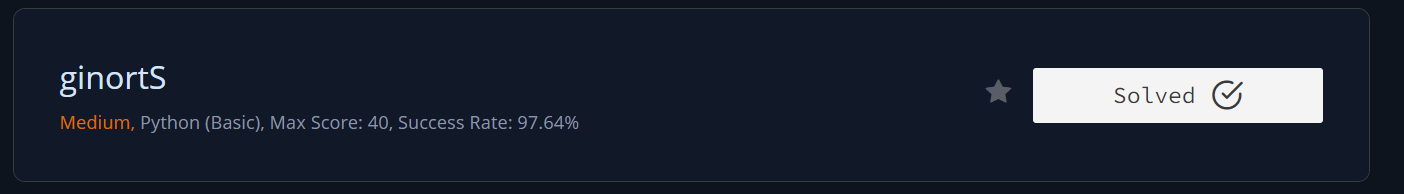
    elif int(i) % 2 !=0:

        odd+= i

    elif int(i) % 2 == 0:

        even += i

print(lower+upper+odd+even)



## 1.2.17 Validating Email Addresses With a Filter.

def fun(s):

    # return True if s is a valid email, else return False

    try:

        username, other = s.split("@")

        websitename, extension = other.split(".")

    except:

        return False

    username = username.replace("-", "").replace("\_", "")

    if username.isalnum() == False:

        return False

    elif websitename.isalnum() == False:

        return False

    elif len(extension) > 3:

        return False

    else:

        return True

def filter\_mail(emails):

    return list(filter(fun, emails))

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    emails = []

    for \_ in range(n):

        emails.append(input())

filtered\_emails = filter\_mail(emails)

filtered\_emails.sort()

print(filtered\_emails)



## 1.2.18 Reduce Function.

from fractions import Fraction

from functools import reduce

def product(fracs):

    t = reduce(lambda a, b : a\*b, fracs)

    return t.numerator, t.denominator

if \_\_name\_\_ == '\_\_main\_\_':

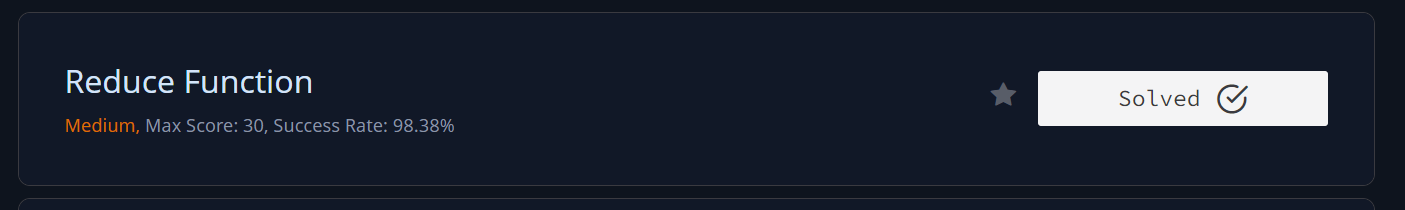
    fracs = []

    for \_ in range(int(input())):

        fracs.append(Fraction(\*map(int, input().split())))

    result = product(fracs)

    print(\*result)



## 1.2.19 Regex Substitution.

import re

pattern = r"(?<= )(&&|\|\|)(?= )"

replacement = lambda x : "and" if x.group() == "&&" else "or"

for i in range (int(input())):

    s = input()

    print(re.sub(pattern, replacement, s))

## C:\Users\mega computers\OneDrive - National University of Sciences & Technology\Desktop\GIT-HUB PYTHON\regex subs.PNG

## 1.2.20 Validating Credit Card Numbers.

# Enter your code here. Read input from STDIN. Print output to STDOUT

import re

for \_ in range(int(input())):

    S=input()

    t = re.search(r"^[456]\d{15}$|^[456]\d{3}-\d{4}-\d{4}-\d{4}$",S)

    if (t):

        if(re.search(r"(\d)\1{3,}|(\d)\2{1}-(\d)\2{1}|-(\d)\4{3}-", S)):

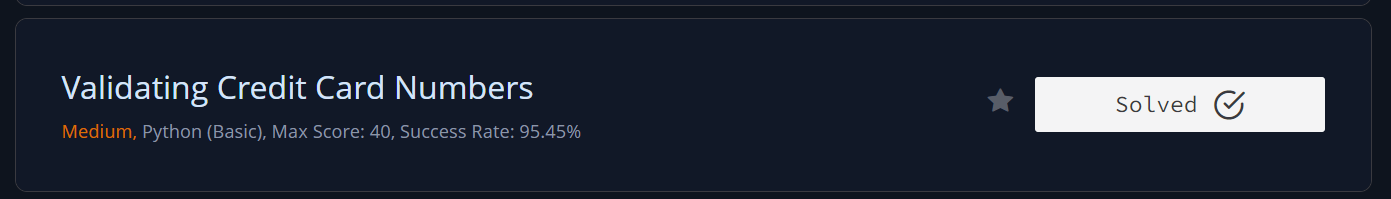
            print("Invalid")

        else:

            print("Valid")

    else:

        print("Invalid")



## 1.2.21 Word Score.

def is\_vowel(letter):

    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score\_words(words):

    score = 0

    for word in words:

        num\_vowels = 0

        for letter in word:

            if is\_vowel(letter):

                num\_vowels += 1

        if num\_vowels % 2 == 0:

            score += 2

        else:

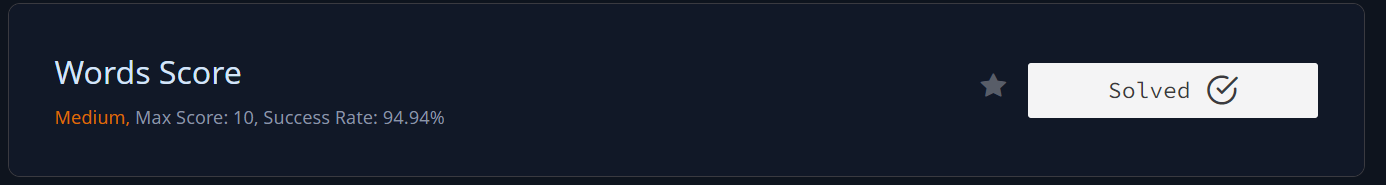
            score += 1

    return score

n = int(input())

words = input().split()

print(score\_words(words))



## 1.2.22 Default Arguments.

class EvenStream(object):

    def \_\_init\_\_(self):

        self.current = 0

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

class OddStream(object):

    def \_\_init\_\_(self):

        self.current = 1

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

def print\_from\_stream(n, stream=EvenStream()):

    stream.\_\_init\_\_()

    for \_ in range(n):

        print(stream.get\_next())

queries = int(input())

for \_ in range(queries):

    stream\_name, n = input().split()

    n = int(n)

    if stream\_name == "even":

        print\_from\_stream(n)

    else:

        print\_from\_stream(n, OddStream())

