ASSIGNMENT 3

Hacker Rank Problem Solutions

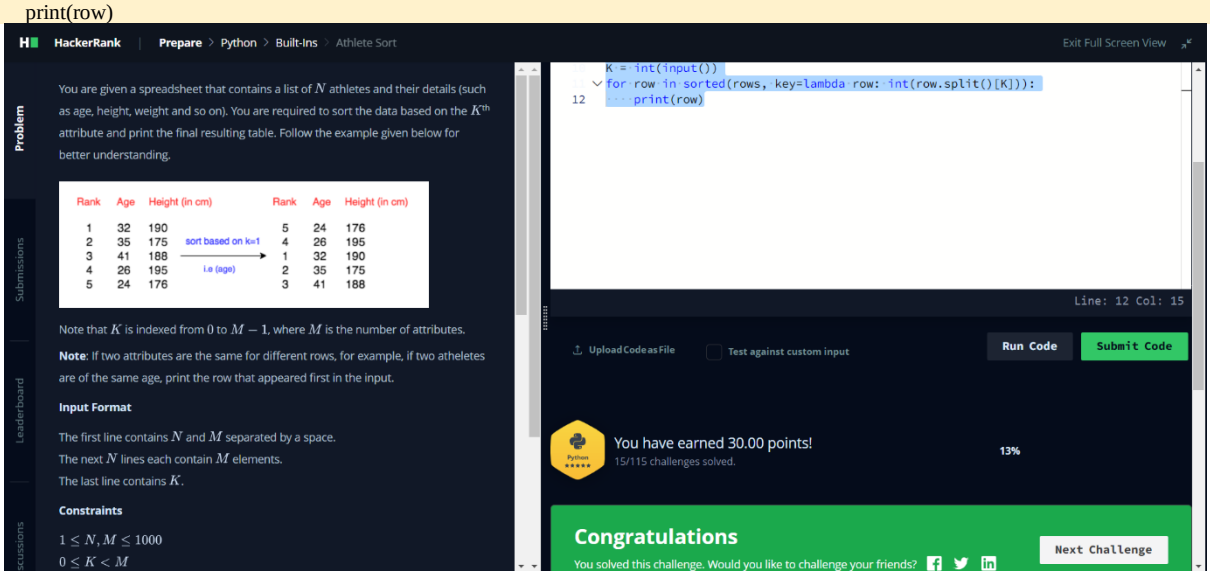
AQIB HABIB MEMON
Reg:450062

Contents

1 Medium Challenges

1.1 Athlete Sort

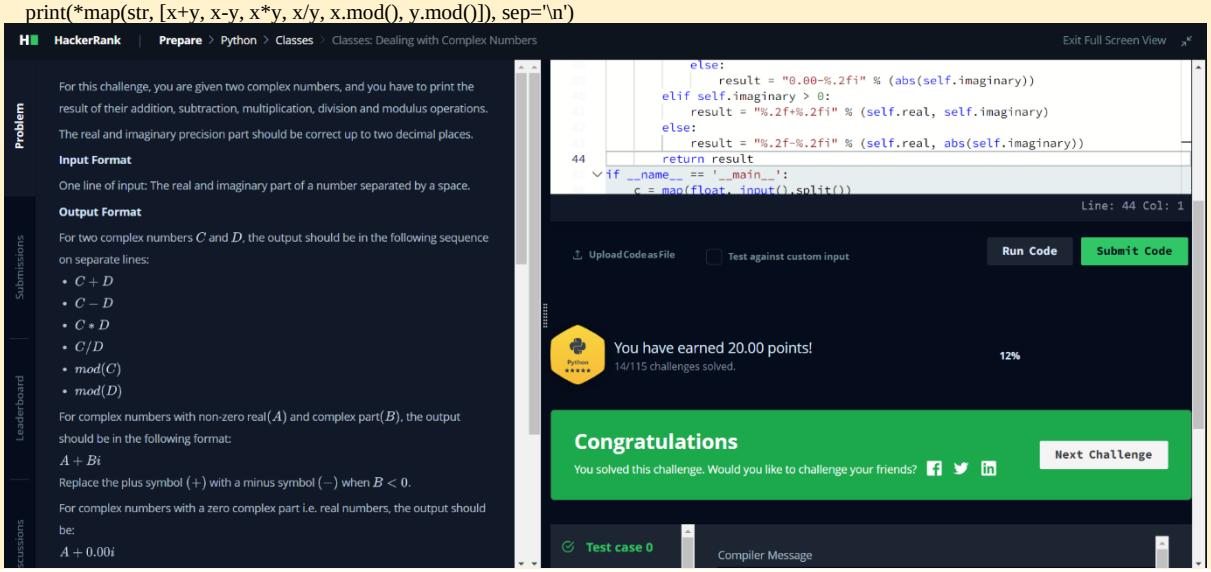
```
import math
import os
import random
import re
import sys
N, M = map(int, input().split())
rows=[]
for _ in range(N):
    rows.append(input())
K = int(input())
for row in sorted(rows, key=lambda row: int(row.split()[K])):
    print(row)
```



1.2 Classes Dealing with Complex Numbers

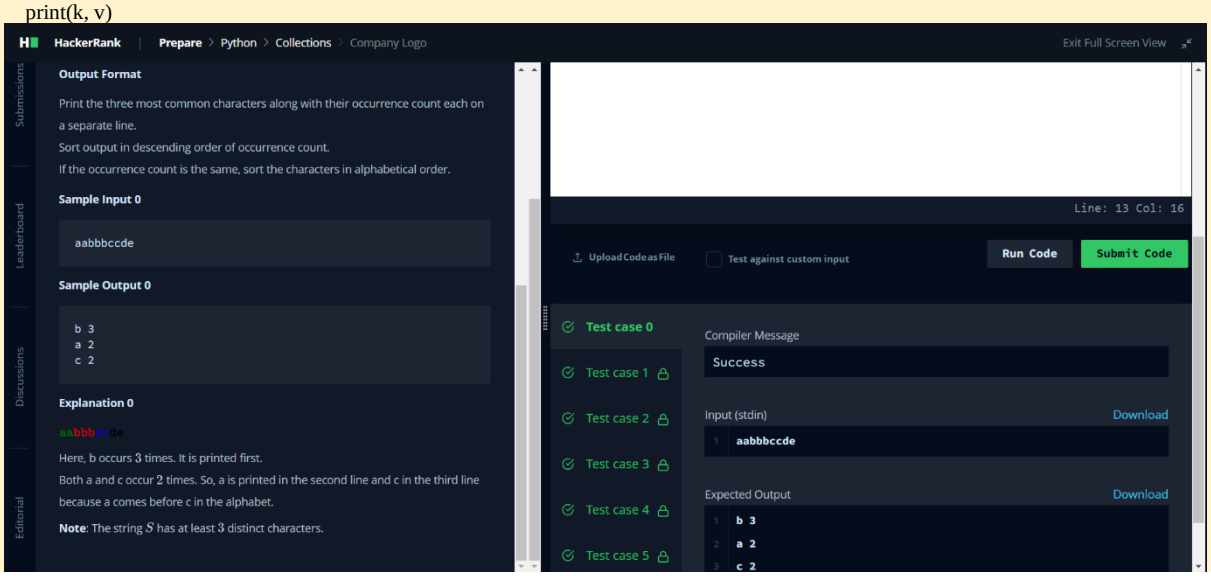
```
import math

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+no.real*self.imaginary)
        return Complex(r, i)

    def __truediv__(self, no):
        conjugate = Complex(no.real, (-no.imaginary))
        num = self*conjugate
        denom = no*conjugate
        try:
            return Complex((num.real/denom.real), (num.imaginary/denom.real))
        except Exception as e:
            print(e)

    def mod(self):
        m = math.sqrt(self.real**2+self.imaginary**2)
        return Complex(m, 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')
```



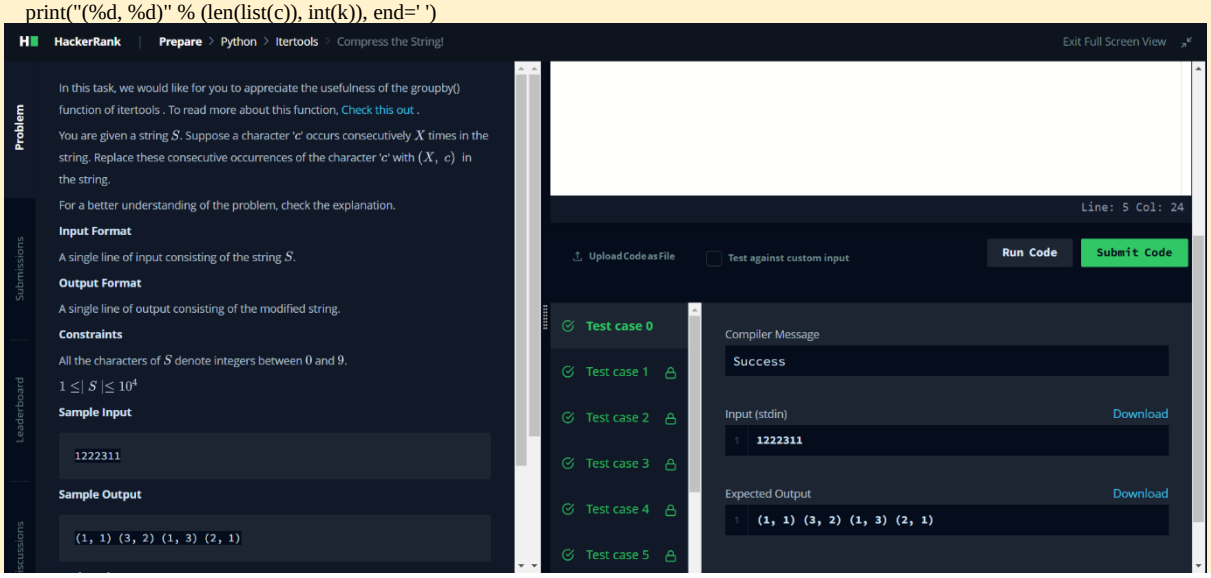
1.3 Company Logo

```
import math
import os
import random
import re
import sys
from collections import Counter

s = input()
s = sorted(s)

Occurence= Counter(list(s))
for k, v in Occurence.most_common(3):
    print(k, v)
```



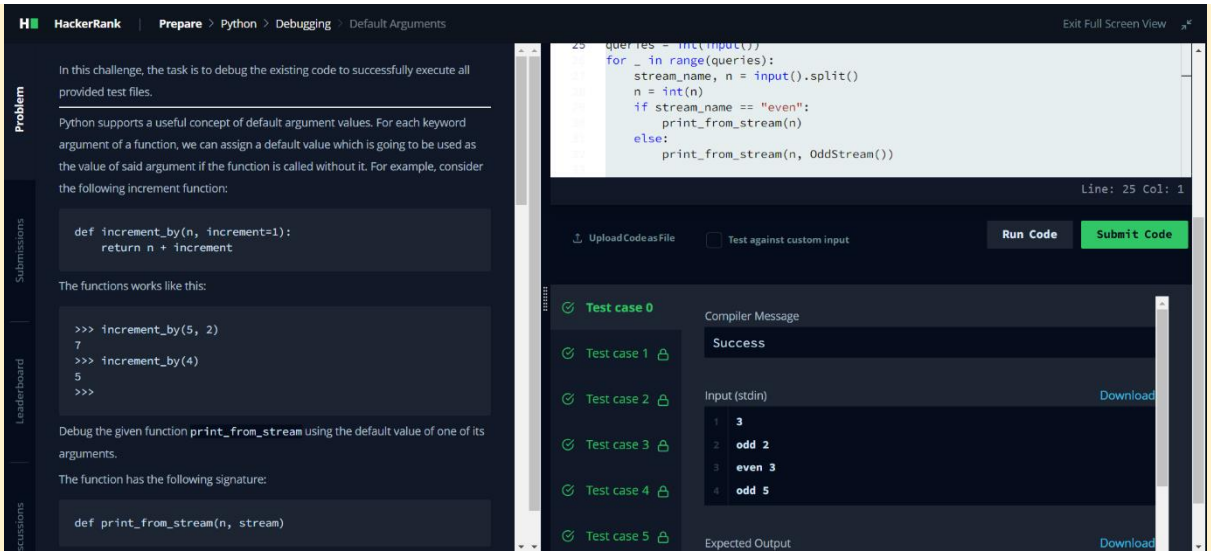
1.4 Compress the String

```
from itertools import groupby
str_in=input()

for k, c in groupby(str_in):
    print("(%d, %d)" % (len(list(c)), int(k)), end=' ')
```



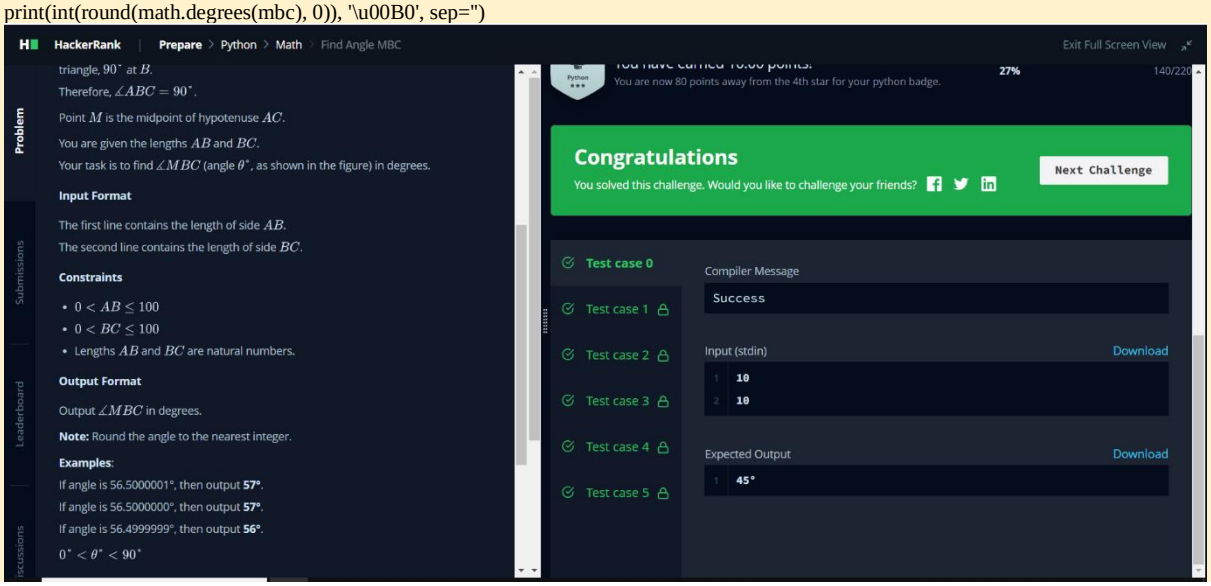
1.5 Default Arguments

```
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())
```



1.6 Find Angle MBC

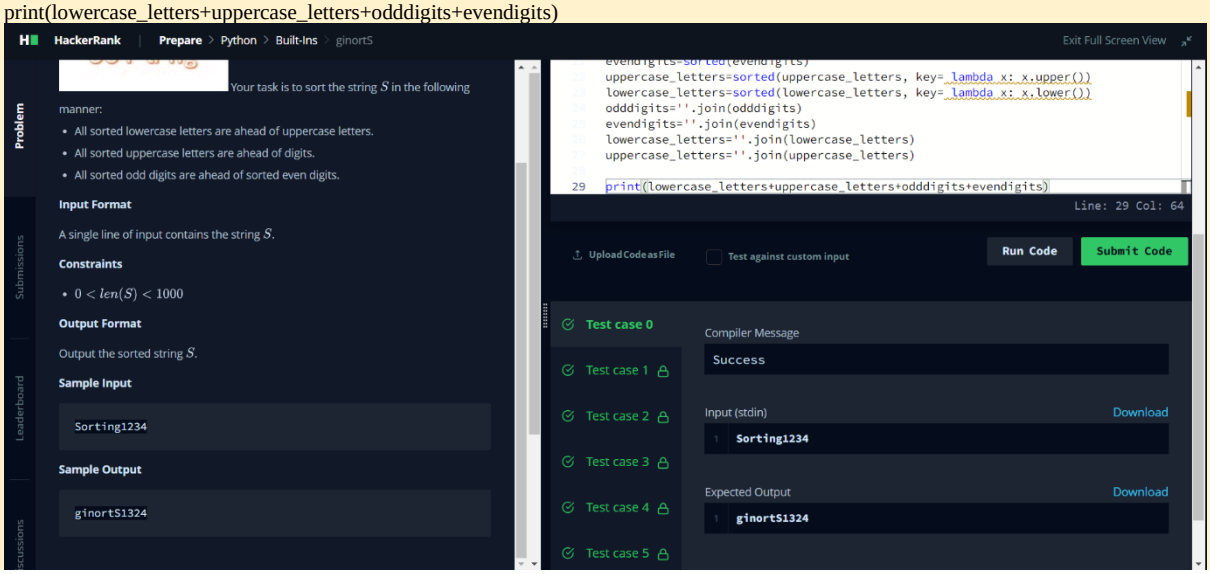
```
import math
ab=int(input())
bc=int(input())
ca=math.hypot(ab,bc)
mc=ca/2
bca=math.asin(1*ab/ca)
bm=math.sqrt((bc**2+mc**2)-(2*bc*mc*math.cos(bca)))
mbc=math.asin(math.sin(bca)*mc/bm)
print(int(round(math.degrees(mbc), 0)), '\u00B0', sep='')
```



1.7 ginortS

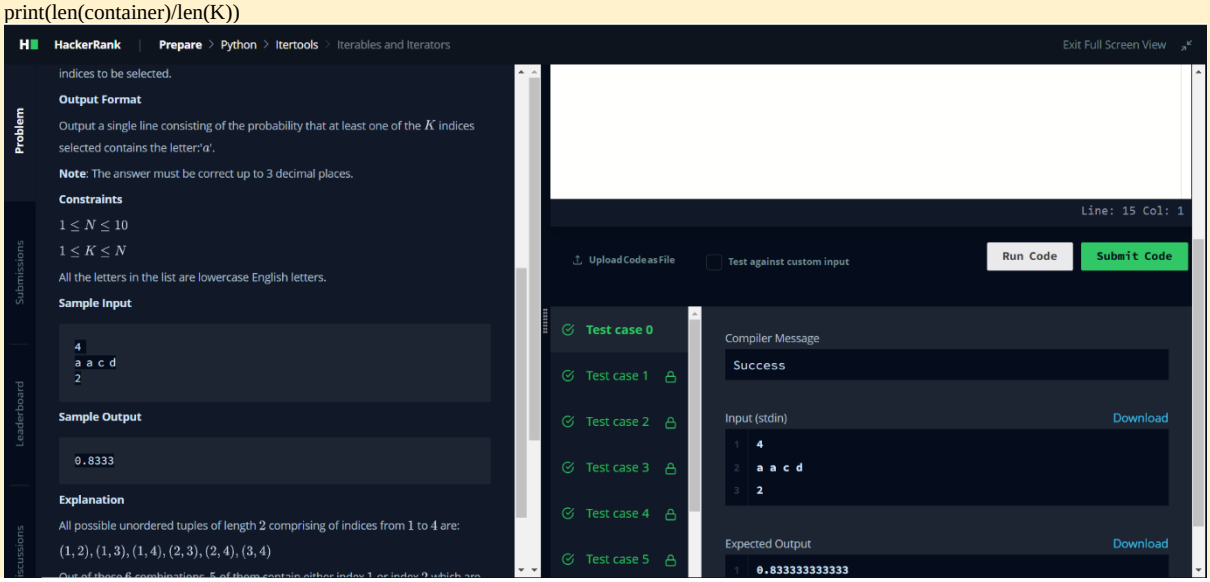
```
odddigits = []
evendigits=[]
uppercase_letters = []
lowercase_letters = []

S=input()

for char in S:
    if char.isdigit():
        if int(char) % 2 == 0:
            evendigits.append(char)
        elif int(char) % 2 !=0:
            odddigits.append(char)
    elif char.isupper():
        uppercase_letters.append(char)
    elif char.islower():
        lowercase_letters.append(char)
```

```
odddigits=sorted(odddigits)
evendigits=sorted(evendigits)
uppercase_letters=sorted(uppercase_letters, key= lambda x: x.upper())
lowercase_letters=sorted(lowercase_letters, key= lambda x: x.lower())
odddigits=''.join(odddigits)
evendigits=''.join(evendigits)
lowercase_letters=''.join(lowercase_letters)
uppercase_letters=''.join(uppercase_letters)

print(lowercase_letters+uppercase_letters+odddigits+evendigits)
```



1.8 Iterables and Iterators

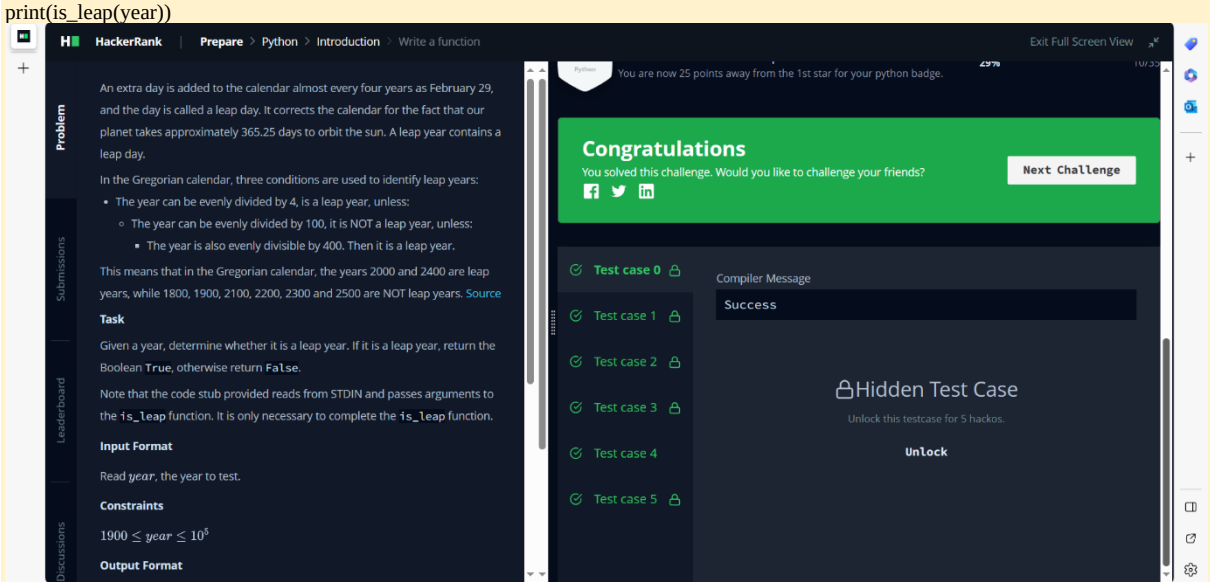
```
from itertools import combinations


N=int(input())
M=list(input().split(" "))
L=int(input())
K=list(combinations(M, L))
container= [i for i in K if "a" in i]
print(len(container)/len(K))
```



1.9 Leap Year

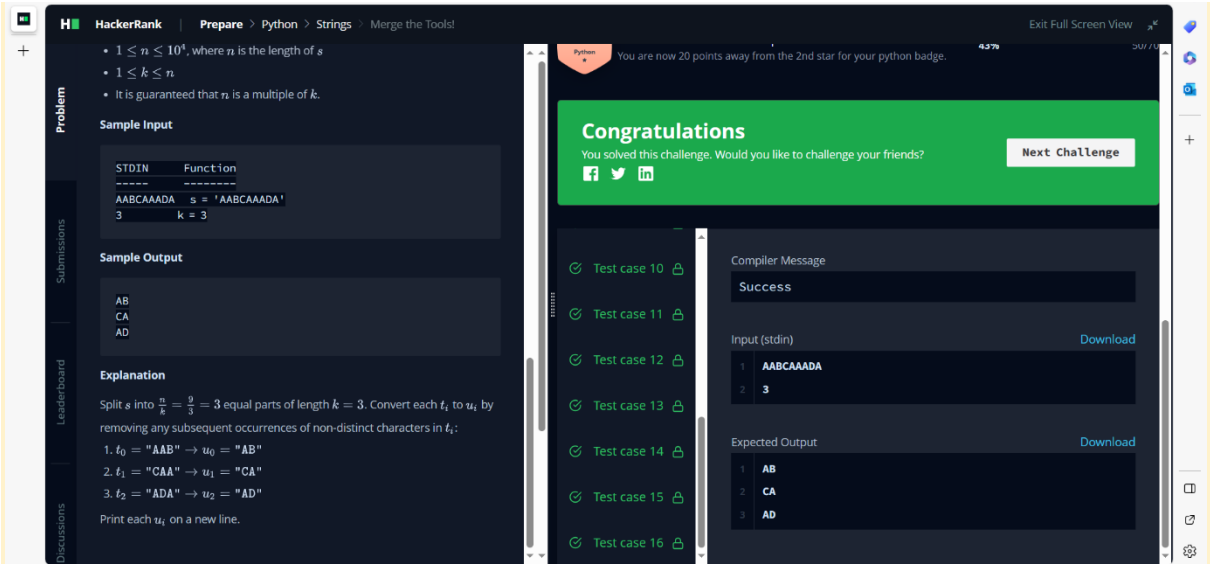
```
def is_leap(year):
    leap = False


    if (year % 4 == 0):
        leap = True
    if (year % 100 == 0):
```

```
        leap = False
    if (year % 400 == 0):
        leap = True

    return leap

year = int(input())
print(is_leap(year))
```



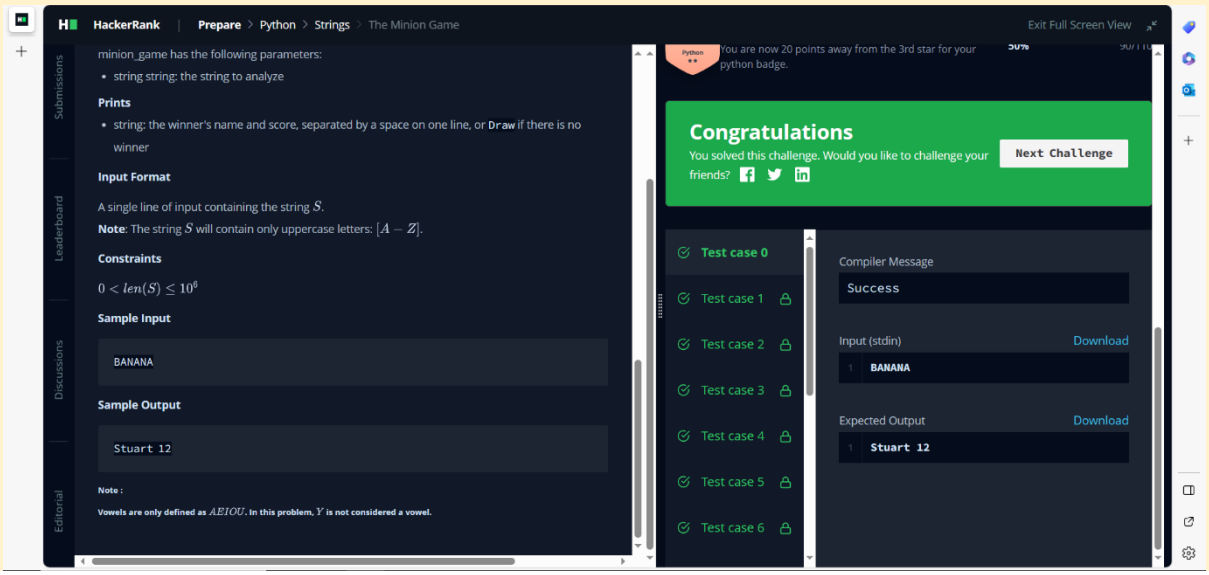
1.10 Merge The Tools

```
def merge_the_tools(string, k):
    # your code goes here
    ls =len(string)
    result = []
    for i in range(0, ls, k):
        result.append(string[i:i+k])
    result_nc=[]
    seen = set()

    for charline in result:
        output_string = ""
        seen = set()
        for char in charline:
            if char not in seen:
                seen.add(char)
                output_string += char
        result_nc.append(output_string)

    for j in result_nc:
        print(j)



if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```



1.11 Minion Game

```
def minion_game(string):
    vowels = 'aeiouAEIOU'
    Kevin_Score= int(0)
    Stuart_Score= int(0)
    l=int(len(string))

    for i in range(l):
        if string[i] in vowels:
            Kevin_Score+= l-i
        else:
            Stuart_Score+= l-i
    if Stuart_Score>Kevin_Score:
        print("Stuart", Stuart_Score)
    elif Stuart_Score==Kevin_Score:
        print("Draw")
    else:
        print("Kevin", Kevin_Score)



if __name__ == '__main__':
    s = input()
    minion_game(s)
```



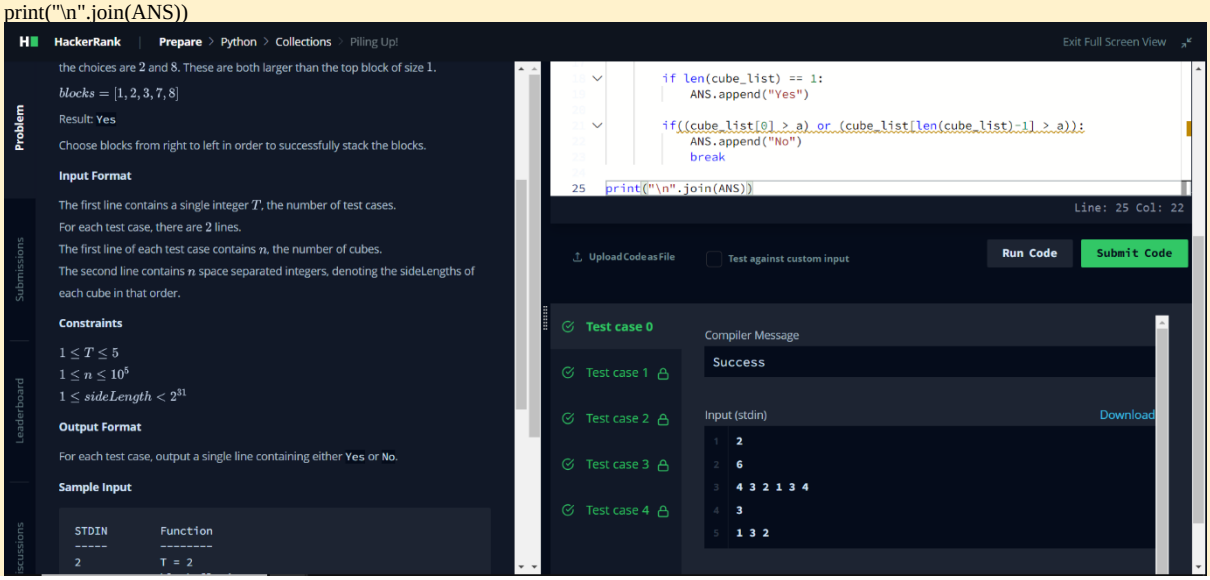
1.12 No Idea

```
happiness = 0
```

```
n, m = map(int, input().split(' '))
arr = list(map(int, input().split(' ')))
good_bhai = set(map(int, input().split(' ')))
bad_bhai = set(map(int, input().split(' ')))

for i in arr:
    if i in good_bhai:
        happiness += 1
    elif i in bad_bhai:
        happiness -= 1
print(happiness)
```

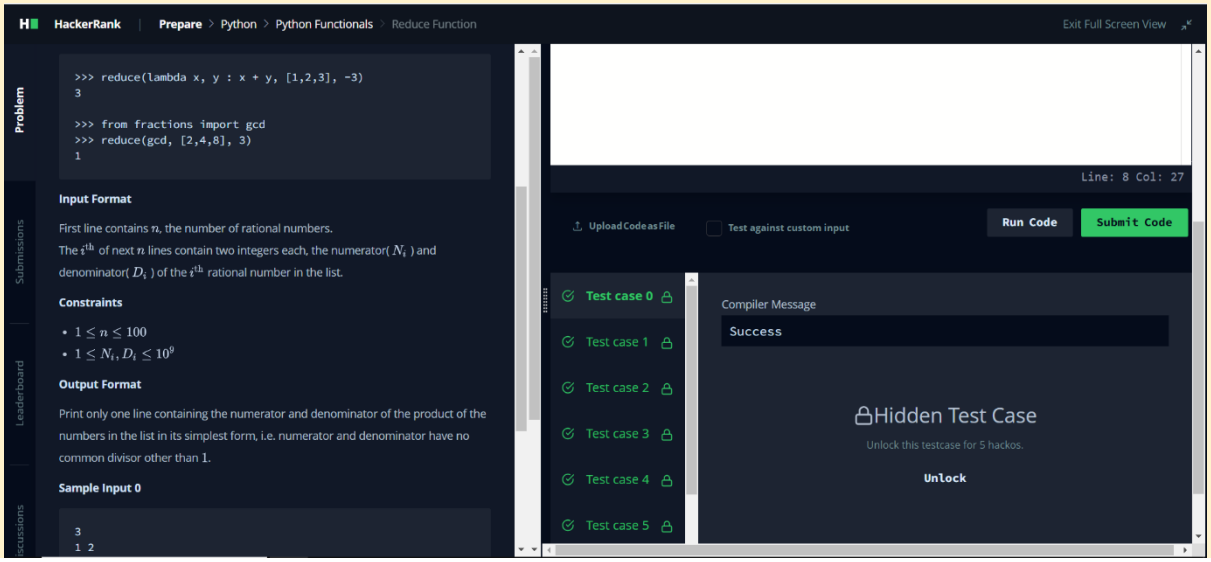
1.13 Piling Up

```
Cube = int(input())
ANS = []
for _ in range(Cube):
    n = int(input())
    cube_list = list(map(int, input().split()))

    for _ in range(n-1):
        if cube_list[0] >= cube_list[len(cube_list)-1]:
            a = cube_list[0]
            cube_list.pop(0)
        elif cube_list[0] < cube_list[len(cube_list)-1]:
            a = cube_list[len(cube_list)-1]
            cube_list.pop(len(cube_list)-1)
        else:
            pass

        if len(cube_list) == 1:
            ANS.append("Yes")

        if((cube_list[0] > a) or (cube_list[len(cube_list)-1] > a)):
            ANS.append("No")
            break

print("\n".join(ANS))
```



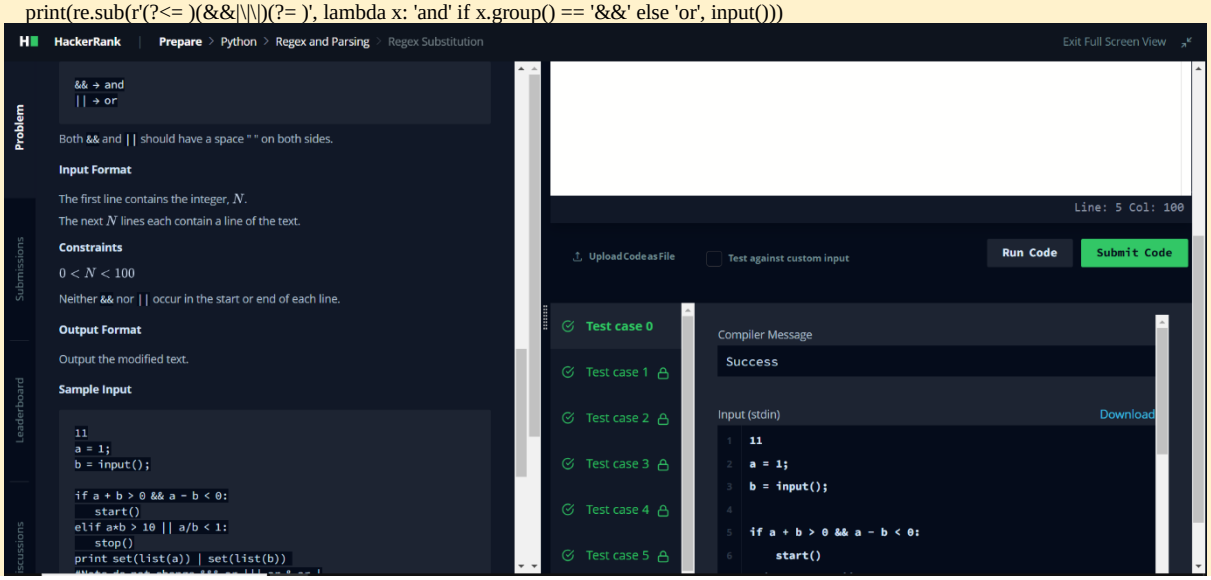
1.14 Reduce Function

```
from fractions import Fraction
from functools import reduce

def product(fracs):
    t = reduce(lambda x, y: x * y, 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)
```



1.15 Regex and Parsing

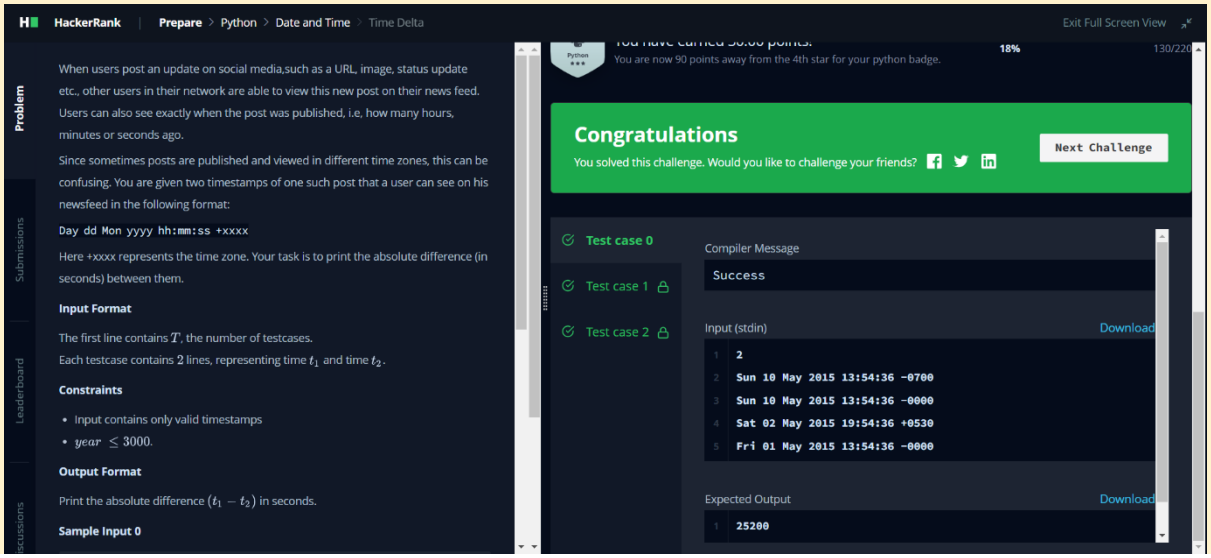
```
import re
for _ in range(int(input())):
    print(re.sub(r'(?<= )(&&|\|\|)(?= )', lambda x: 'and' if x.group() == '&&' else 'or', input()))
```



1.16 Time Delta

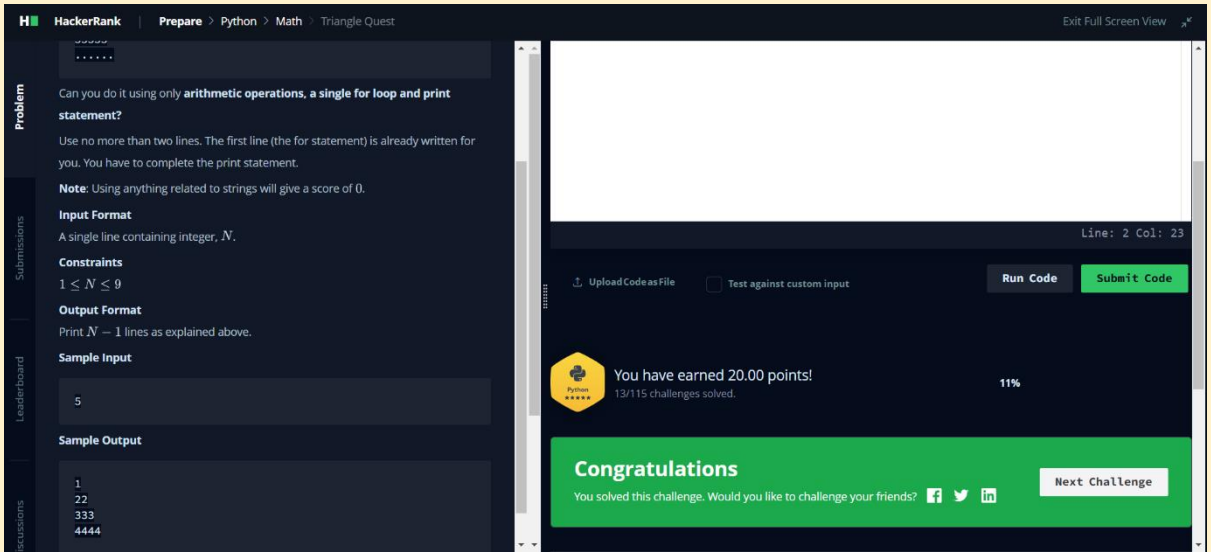
```
import math
import os
import random
import re
import sys

# Complete the time_delta function below.
from datetime import datetime
def time_delta(t1, t2):
    time_format = '%a %d %b %Y %H:%M:%S %z'
    t1 = datetime.strptime(t1, time_format)
    t2 = datetime.strptime(t2, time_format)
    return str(int(abs((t1-t2).total_seconds())))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    t = int(input())

    for t_itr in range(t):
        t1 = input()
```

```
    t2 = input()

    delta = time_delta(t1, t2)

    fptr.write(delta + '\n')

fptr.close()
```



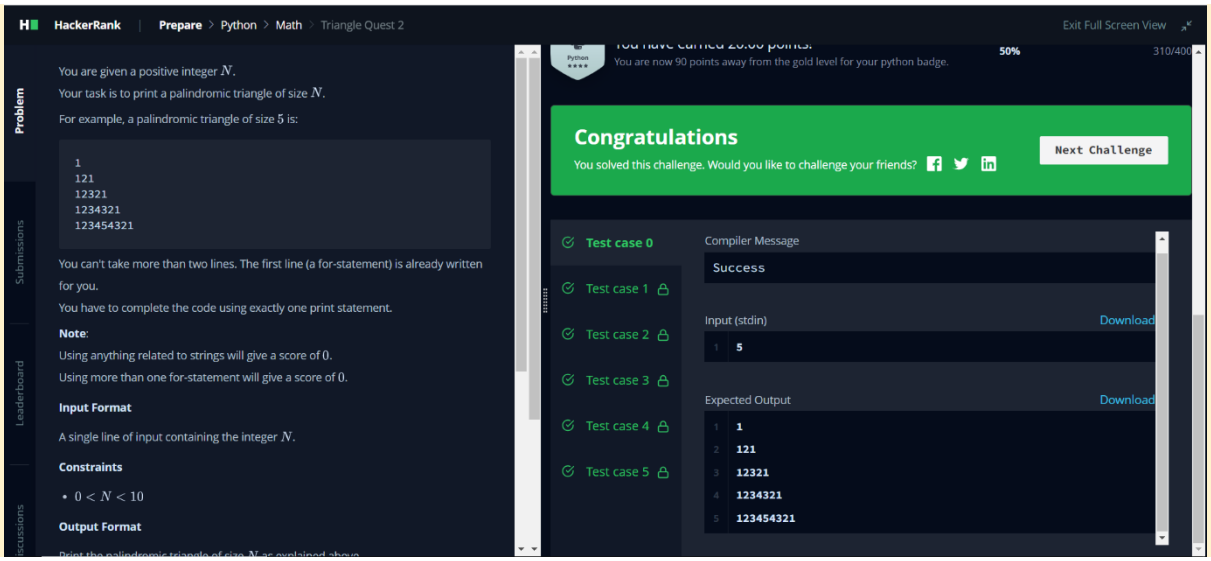
1.17 Triangle Quest

```
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)
```



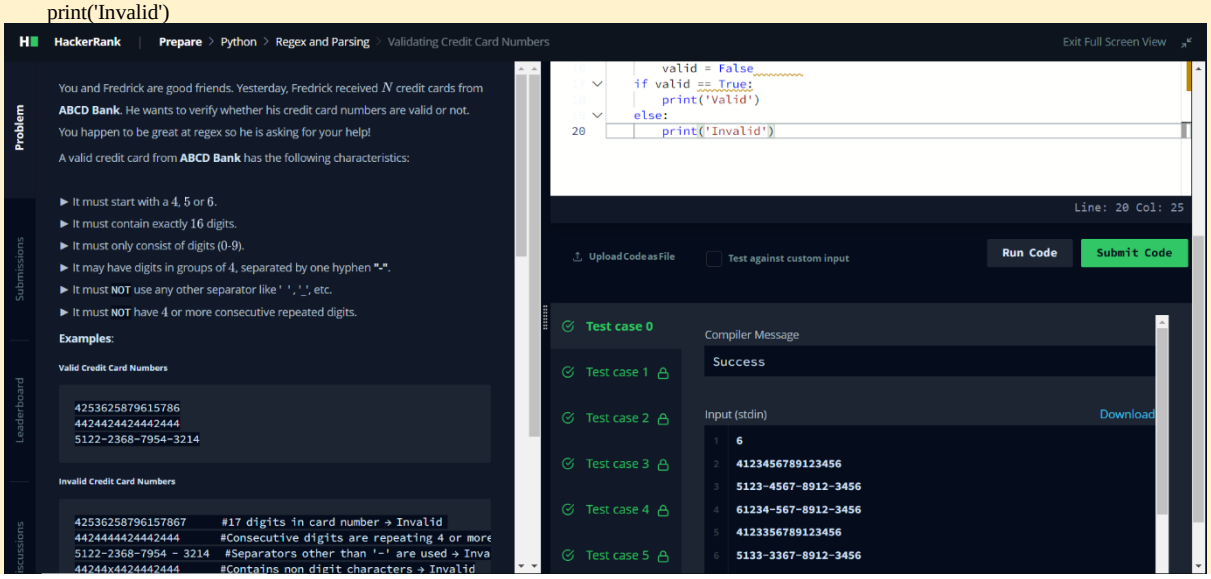
1.18 Triangle Quest 2

```
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)
```



1.19 Validating Credit Card Numbers

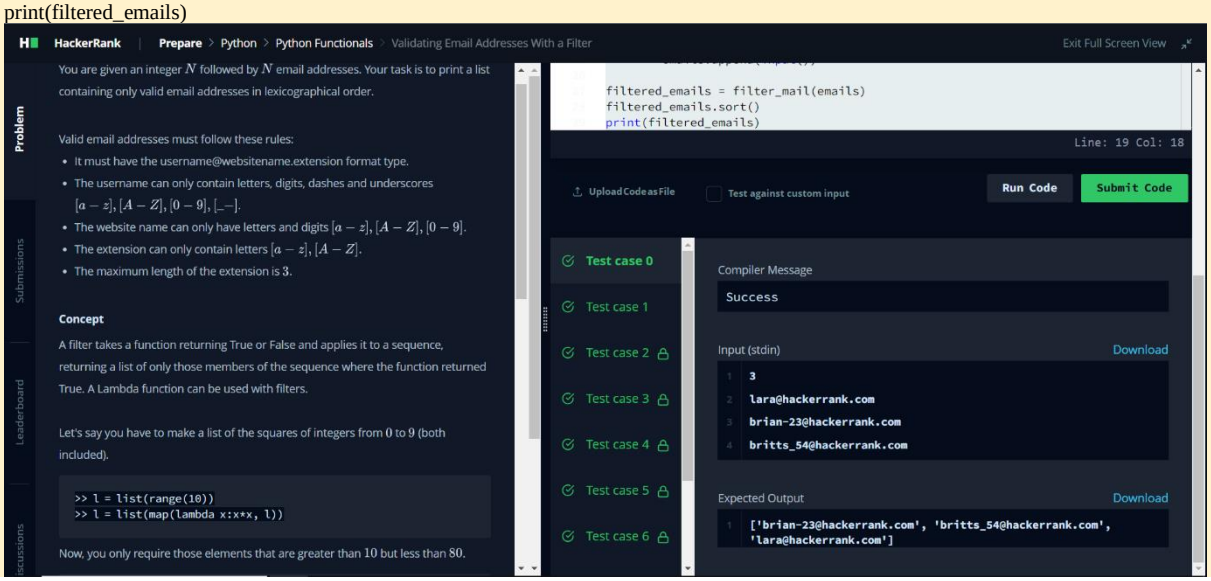
```
import re
n = int(input())
for t in range(n):
    credit = input().strip()
    credit_removed_hiphen = credit.replace('-','')
    valid = True
    length_16 = bool(re.match(r'^[4-6]\d{15}$',credit))
    length_19 = bool(re.match(r'^[4-6]\d{3}-\d{4}-\d{4}-\d{4}$',credit))
    consecutive = bool(re.findall(r'(?=(\d)\1\1\1)',credit_removed_hiphen))
    if length_16 == True or length_19 == True:
        if consecutive == True:

            valid=False
    else:
        valid = False
    if valid == True:
        print('Valid')
    else:
        print('Invalid')
```



1.20 Validating Email Addresses With a Filter

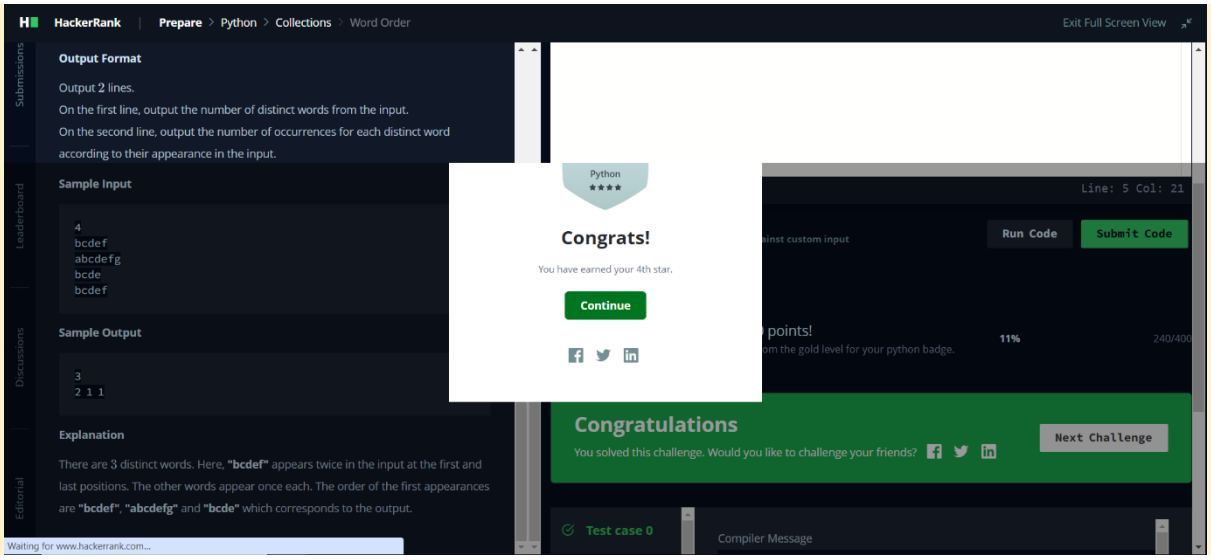
```
def fun(email):
    try:
        username, url = email.split('@')
        website, extension = url.split('.')
    except ValueError:
        return False
    if username.replace('-', '').replace('_', '').isalnum() is False:
```

```
        return False
    elif website.isalnum() is False:
        return False
    elif len(extension) > 3:
        return False
    elif extension.isalpha() is False:
        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)
```



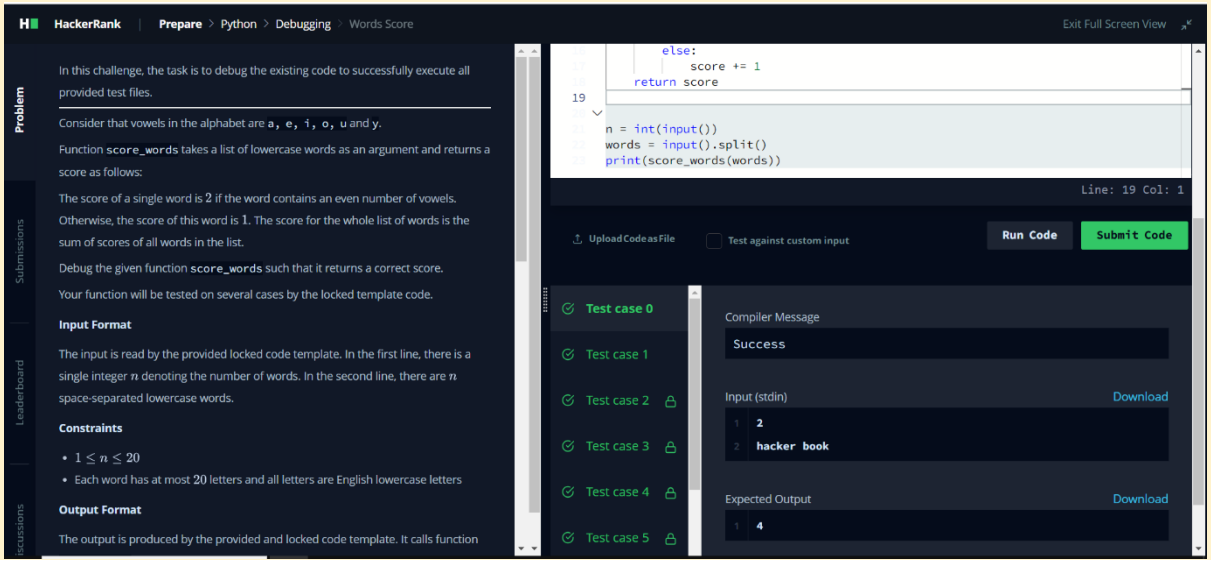
1.21 Word Order

```
from collections import Counter
N = int(input())
words = []
for i in range(N):
    words.append(input().strip())
count = Counter(words)
print(len(count))
print(*count.values())
```



1.22 Word Score

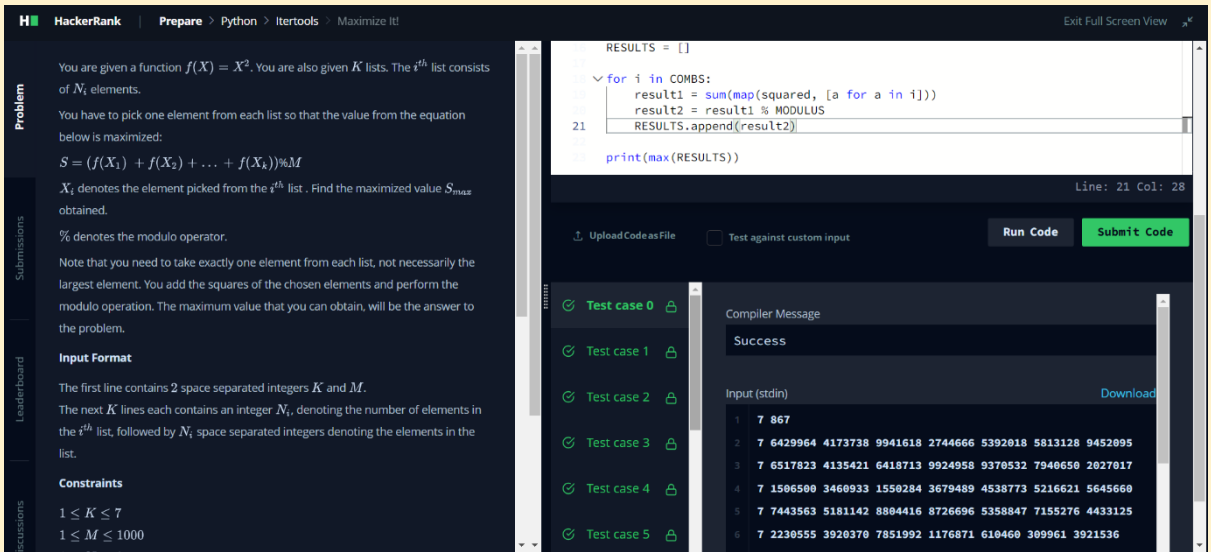
```
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

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))
```



2 Hard Challenges

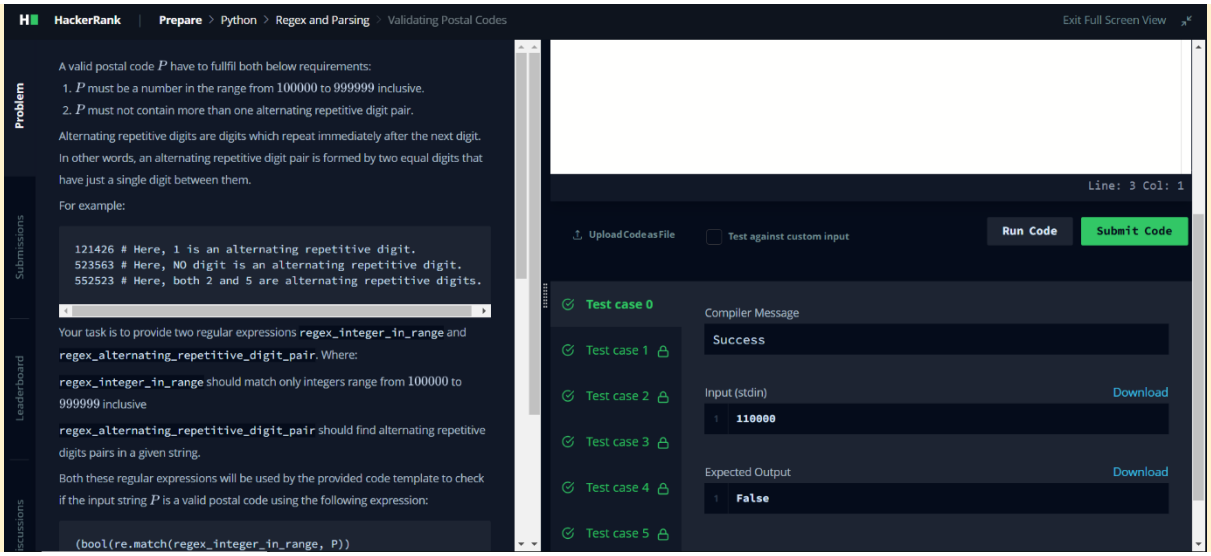
2.1 Maximize It

```
import itertools

NUMBER_OF_LISTS, MODULUS = map(int, input().split())
LISTS_OF_LISTS = []

for i in range(0, NUMBER_OF_LISTS):
    new_list = list(map(int, input().split()))
    del new_list[0]
    LISTS_OF_LISTS.append(new_list)

def squared(element):
    return element**2

COMBS = list(itertools.product(*LISTS_OF_LISTS))
RESULTS = []

for i in COMBS:
    result1 = sum(map(squared, [a for a in i]))
    result2 = result1 % MODULUS
    RESULTS.append(result2)

print(max(RESULTS))
```



2.2 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)
```



2.3 Matrix Script

```
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 = []
character_ar = ['] * (n*m)
for i in range(n):
    line = input()
    for j in range(m):
        character_ar[i+(j*n)]=line[j]
decoded_str = ''.join(character_ar)
final_decoded_str = re.sub(r'(?<=[A-Za-z0-9])([ !@#$%&]+)(?=[A-Za-z0-9])',' ',decoded_str)
print(final_decoded_str)
```