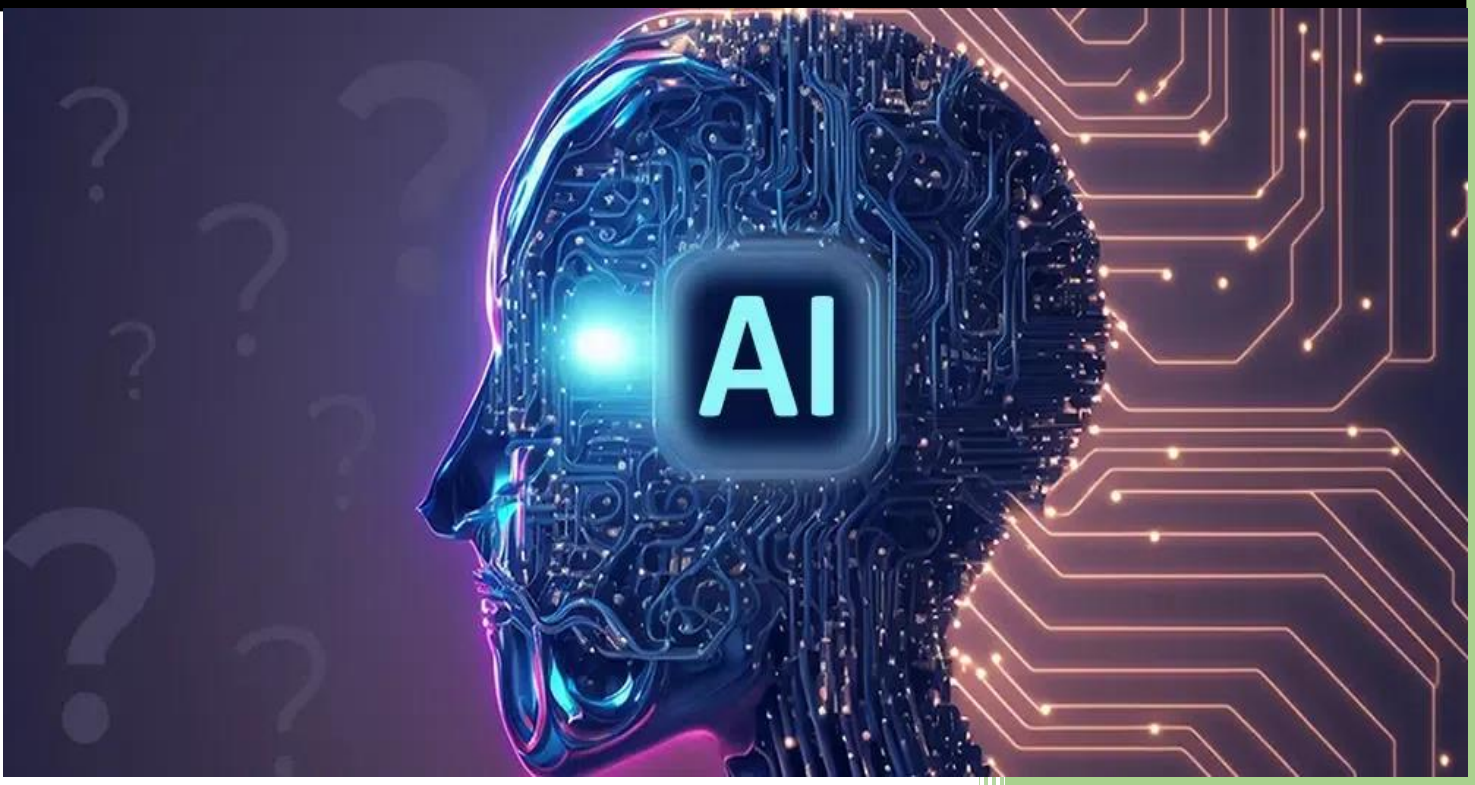
AI Assignment 03

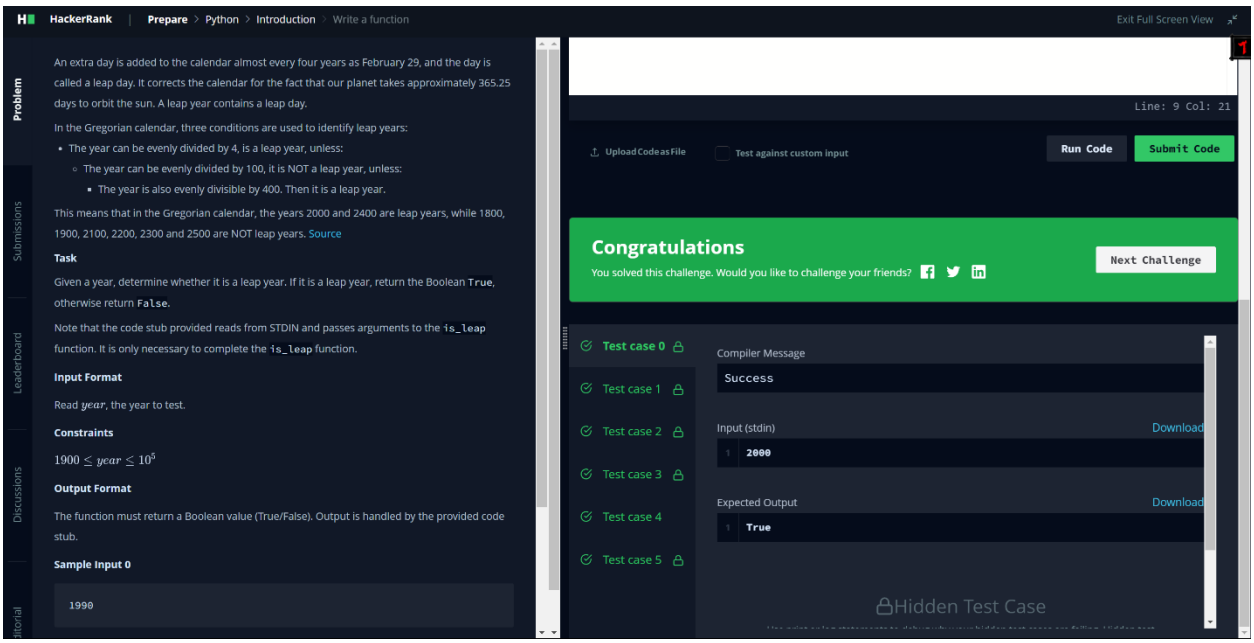


MUHAMMAD ABDULLAH YOUSAF

450767

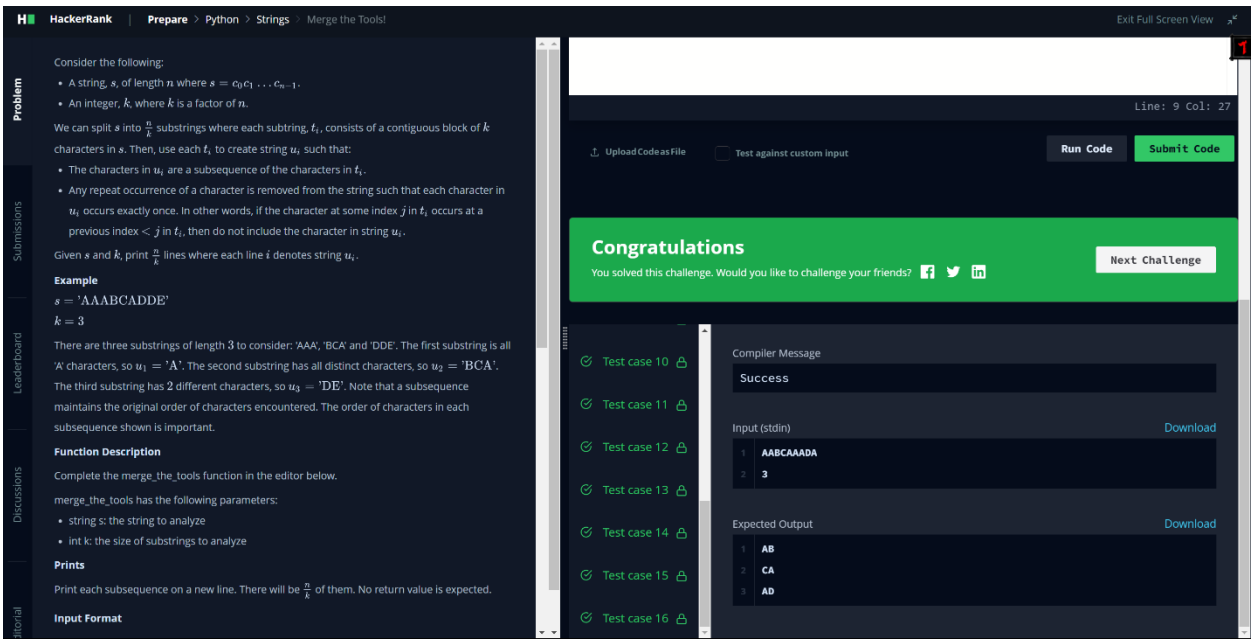
MEDIUM 01

```python
def is_leap(year):
    leap = False
    leap = (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0)
    return leap

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



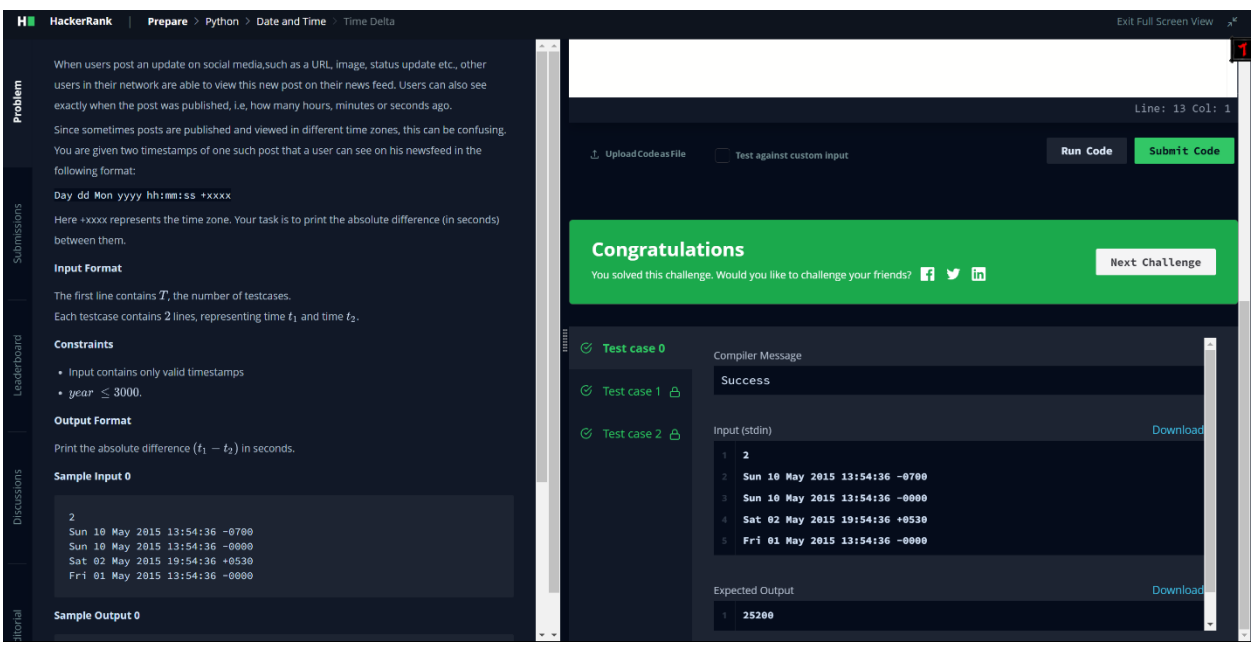
MEDIUM 02

```python
from collections import OrderedDict

def merge_the_tools(string, k):
    # your code goes here
    for i in range(0, len(string)-k+1, k):
        print(''.join(OrderedDict.fromkeys(string[i:i + k])))
if __name__ == '__main__':
```



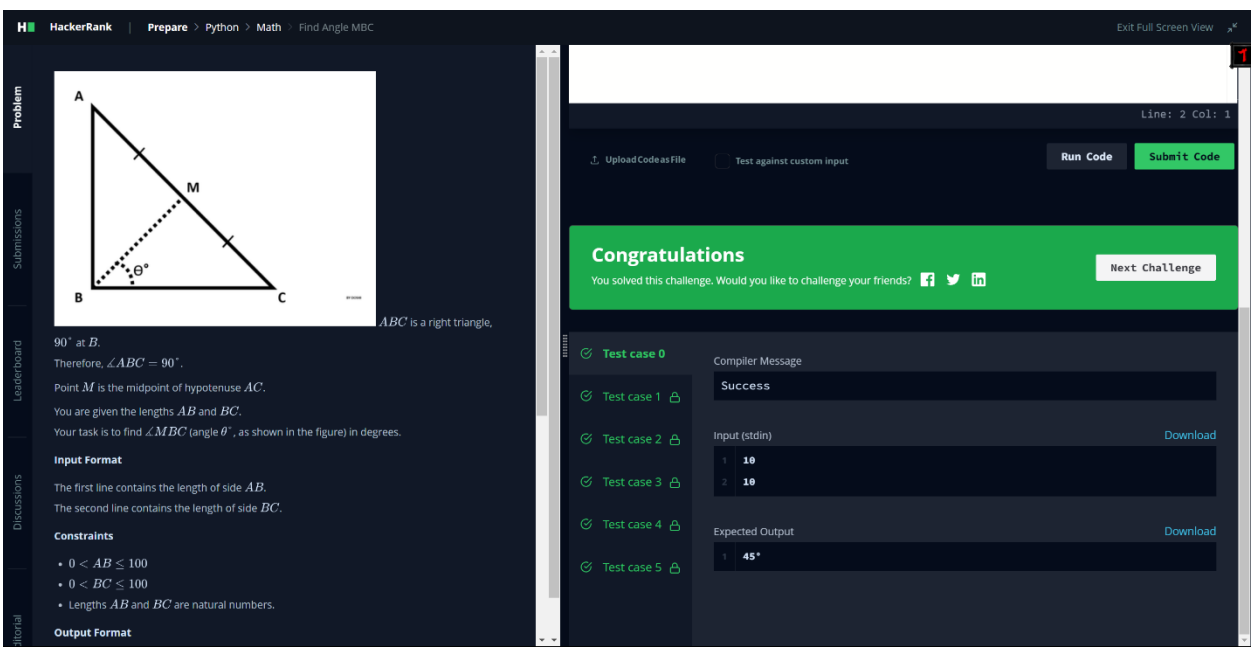
MEDIUM 03

```
import datetime
cas = int(input())
time_format = "%a %d %b %Y %H:%M:%S %z"
for _ in range(cas):
    timestamp1 = input().strip()
    timestamp2 = input().strip()
    time_second1 = datetime.datetime.strptime(timestamp1, time_form
at)
    time_second2 = datetime.datetime.strptime(timestamp2, time_form
at)
    print(int(abs((time_second1 - time_second2).total_seconds())))
```



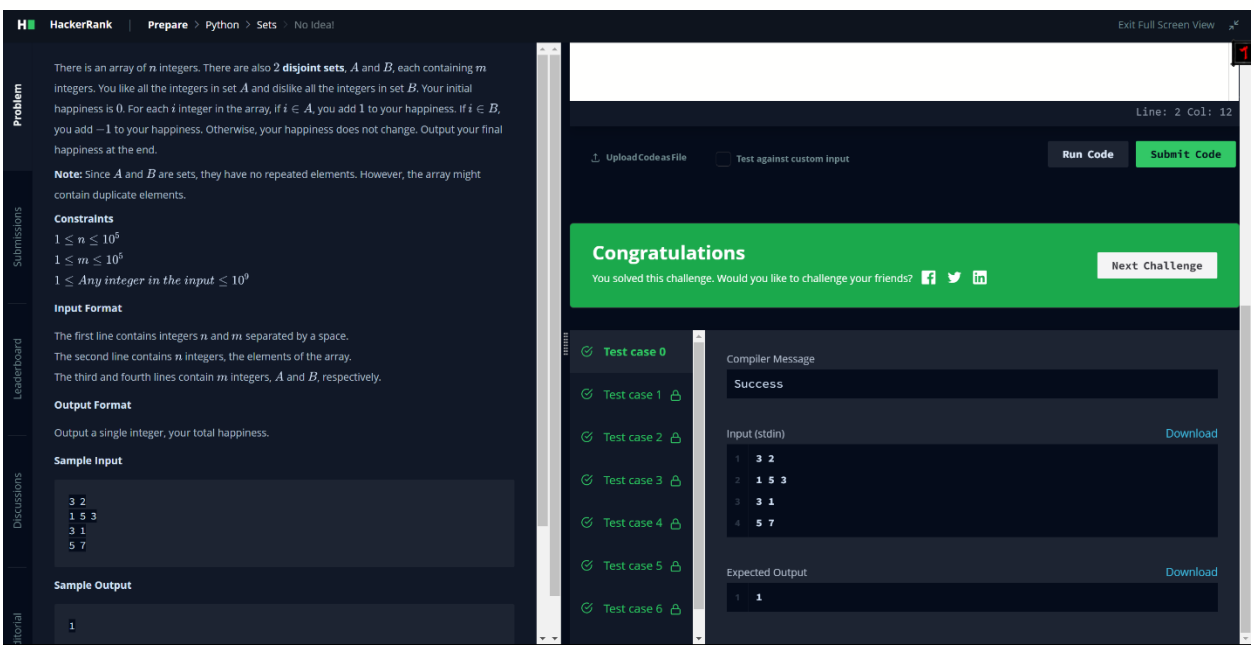
MEDIUM 04

```python
import math

ab = float(input())
bc = float(input())
ac = math.sqrt(ab**2 + bc**2)
bm = ac / 2.0
mc = bm
# let,
b = mc
c = bm
a = bc
angel_b_radian = math.acos(a / (2 * b))
angel_b_degree = int(round((180 * angel_b_radian) / math.pi))
print(f"{angel_b_degree}{chr(176)}")
```



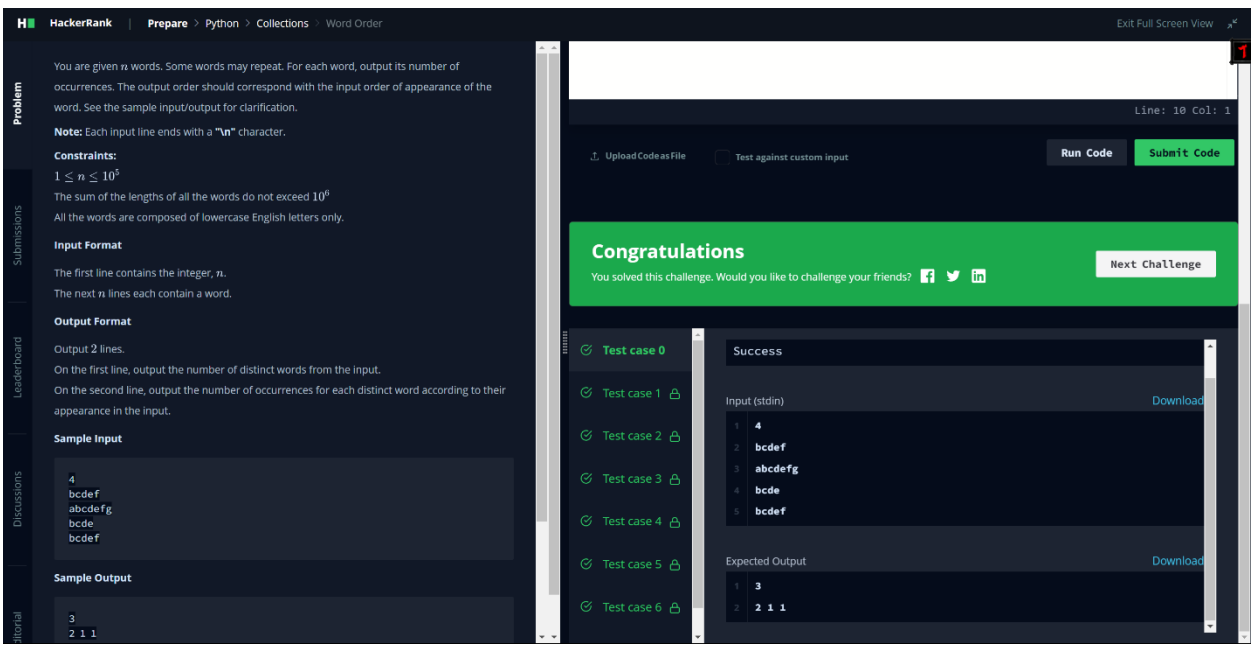
MEDIUM 05

```python
from collections import Counter

n, m = map(int, input().split())
data = list(map(int, input().split()))
data_counter = Counter(data)
data_set = set(data)
set_a = set(map(int, input().split()))
set_b = set(map(int, input().split()))
happiness = 0
for i in data_set & set_a:
    happiness += data_counter[i]
for i in data_set & set_b:
    happiness -= data_counter[i]
print(happiness)
```



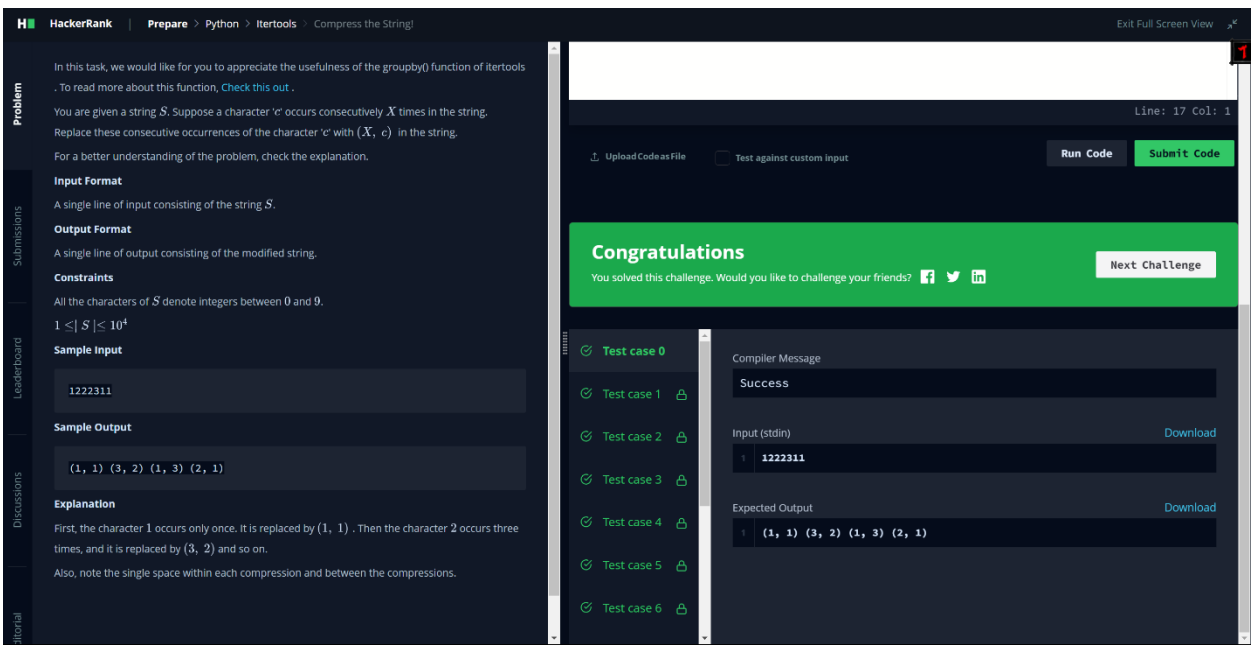
MEDIUM 06

```python
from collections import Counter, OrderedDict


class OrderedCounter(Counter, OrderedDict):
    pass


word_ar = []
n = int(input())
for i in range(n):
    word_ar.append(input().strip())
word_counter = OrderedCounter(word_ar)
print(len(word_counter))
for word in word_counter:
    print(word_counter[word], end=" ")
```



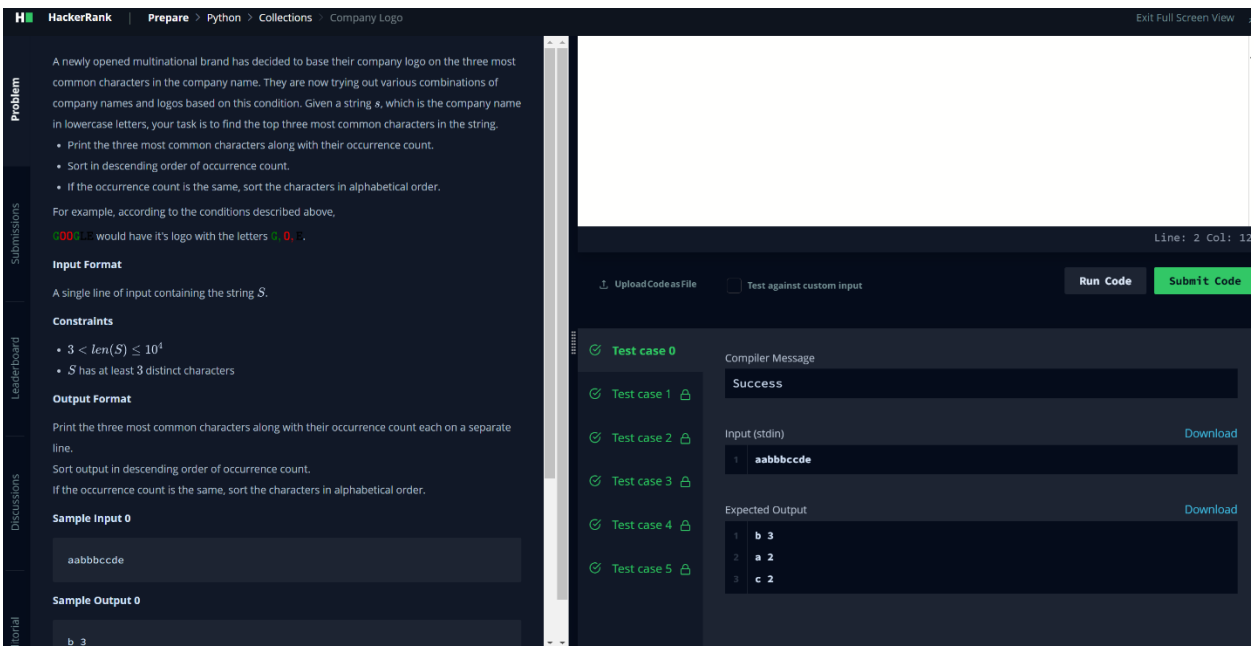
MEDIUM 07

```python
import itertools

s = input().strip()
s_unique_element = list(set(s))
group = []
key = []
for k, g in itertools.groupby(s):
    group.append(list(g))
    key.append(k)
for i in range(len(group)):
    group_length = len(group[i])
    k = int(key[i])
    print((group_length, k), end=" ")
```



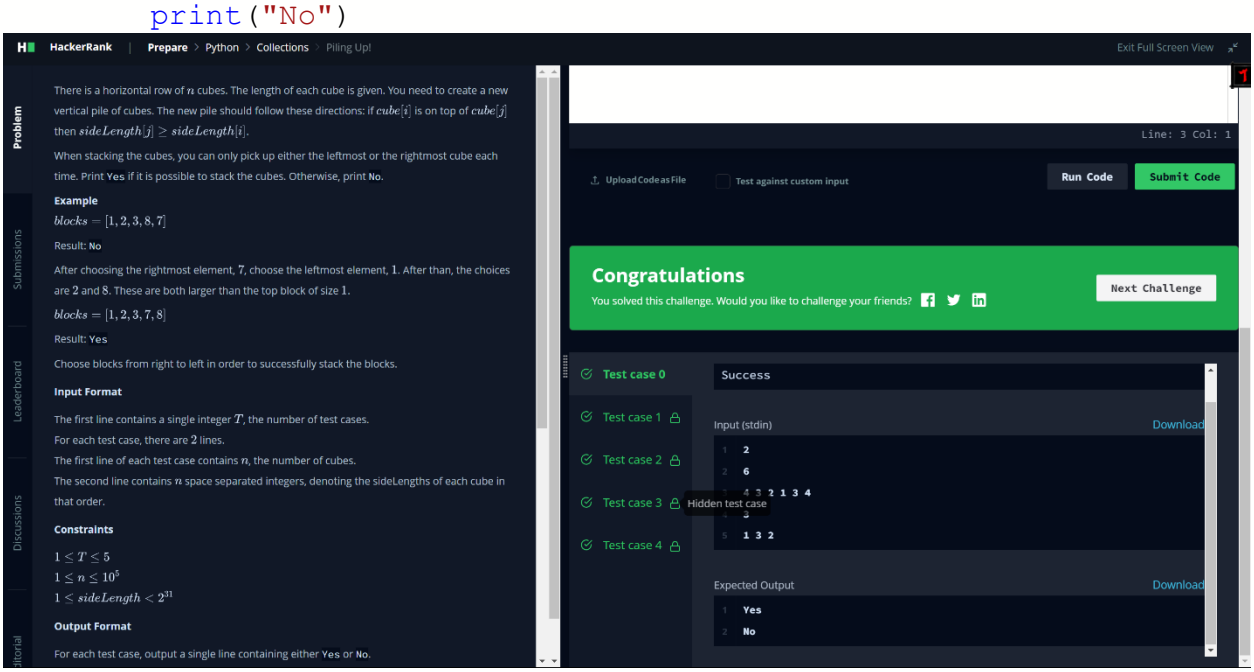
MEDIUM 08

```python
import re
n, m = input().strip().split(' ')
n, m = [int(n), int(m)]
matrix = []
for _ in range(n):
matrix_t = str(input())
matrix.append(matrix_t)
complete = ""
for el in zip(*matrix):
complete += "".join(el)
print(re.sub(r'(?<=\w)([^\w]+)(?=\w)', " ", complete))
```



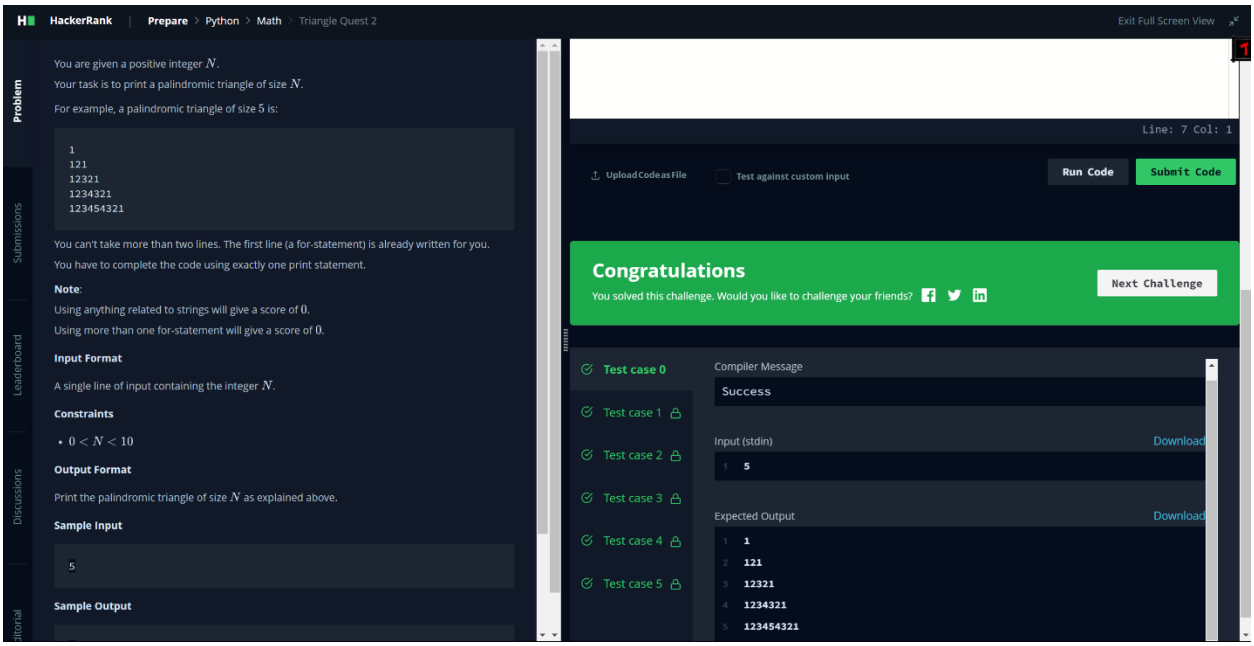
MEDIUM 09

```
from collections import deque

cas = int(input())
for _ in range(cas):
    n = int(input())
    dq = deque(map(int, input().split()))
    possible = True
    element = (2**31) + 1
    while dq:
        left_element = dq[0]
        right_element = dq[-1]
        if left_element >= right_element and element >= left_elemen
t:
            element = dq.popleft()
        elif right_element >= left_element and element >= right_ele
ment:
            element = dq.pop()
        else:
            possible = False
            break
    if possible:
        print("Yes")
    else:
        print("No")
```



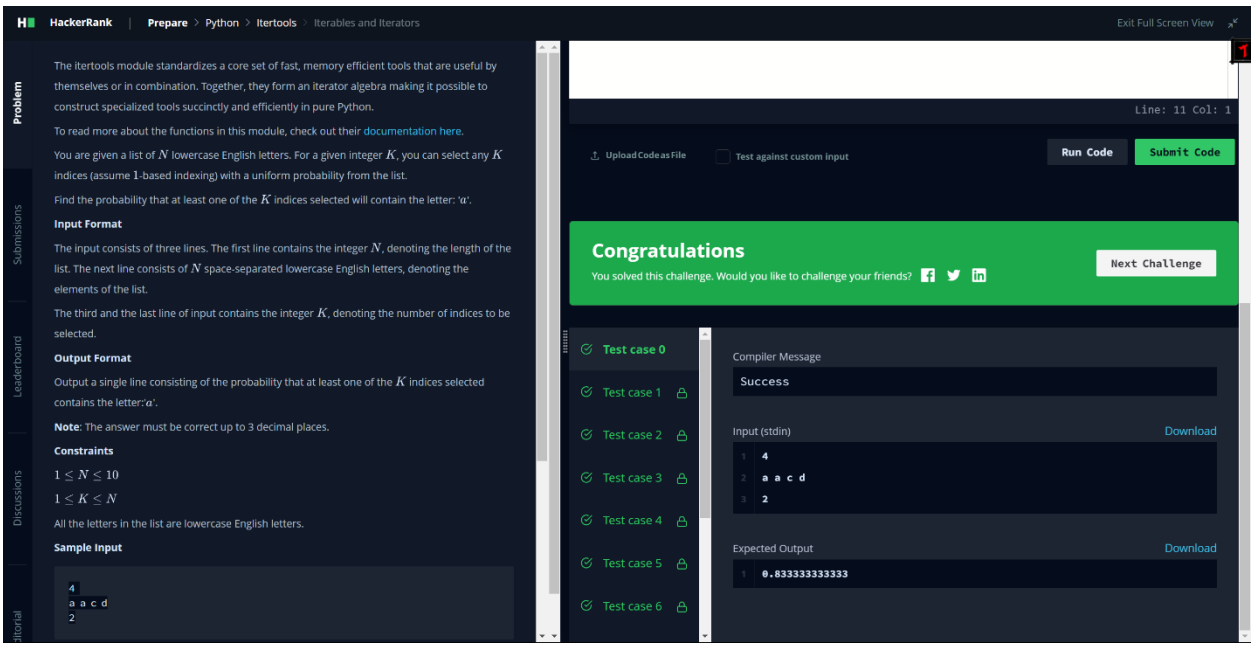
MEDIUM 10

```
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 - 1) // 9) ** 2)
```



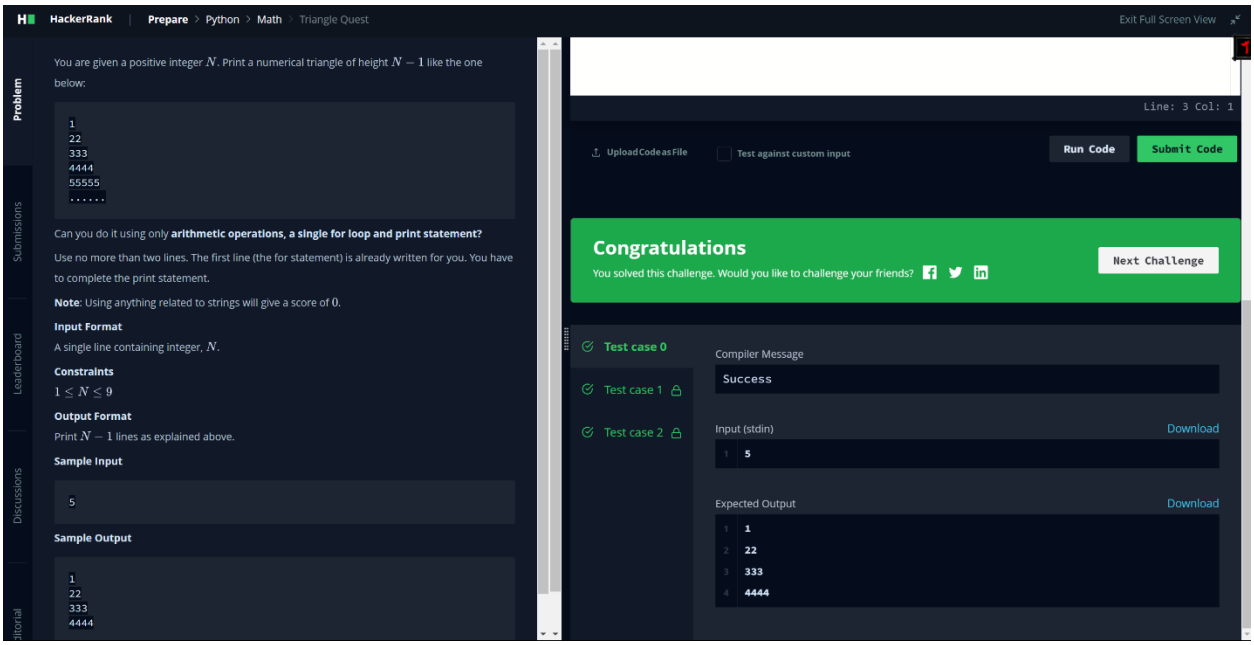
MEDIUM 11

```python
from itertools import combinations

n = int(input())
ar = input().split()
k = int(input())
comb_list = list(combinations(ar, k))
a_list = [e for e in comb_list if "a" in e]
print(len(a_list) / len(comb_list))
```



MEDIUM 12

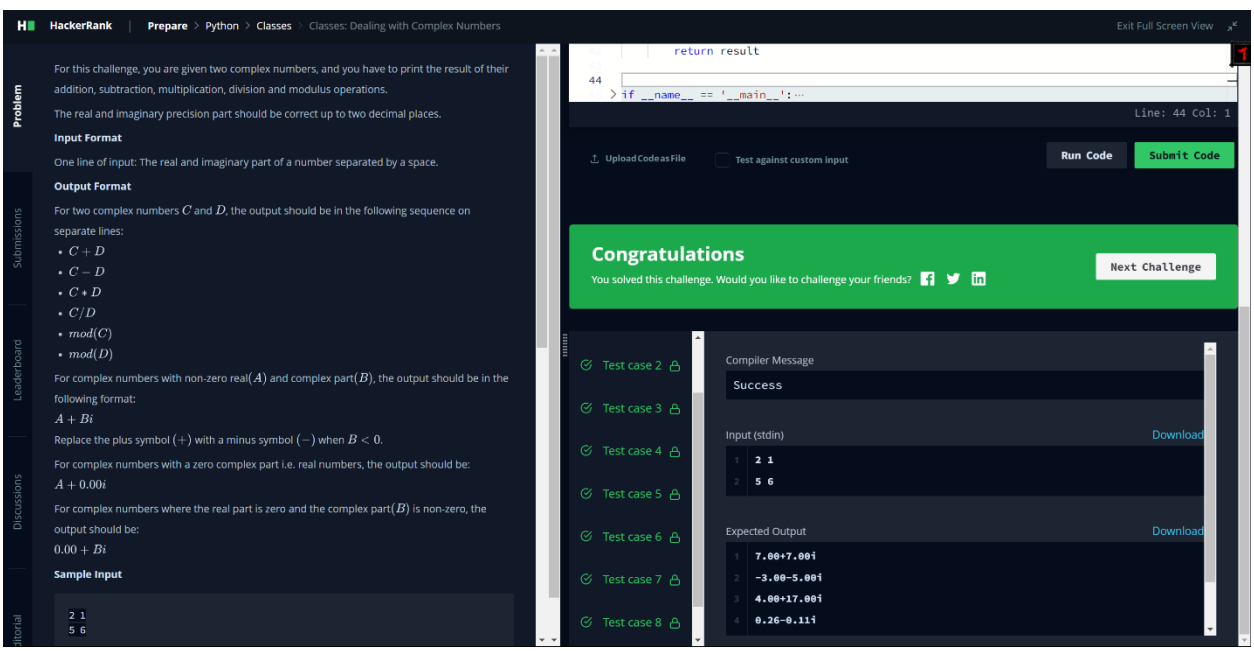
```
for i in range(1,int(input())): #More than 2 lines will result in 0
 score. Do not leave a blank line also
    print(i * ((10**i - 1) // 9))
```



MEDIUM 13

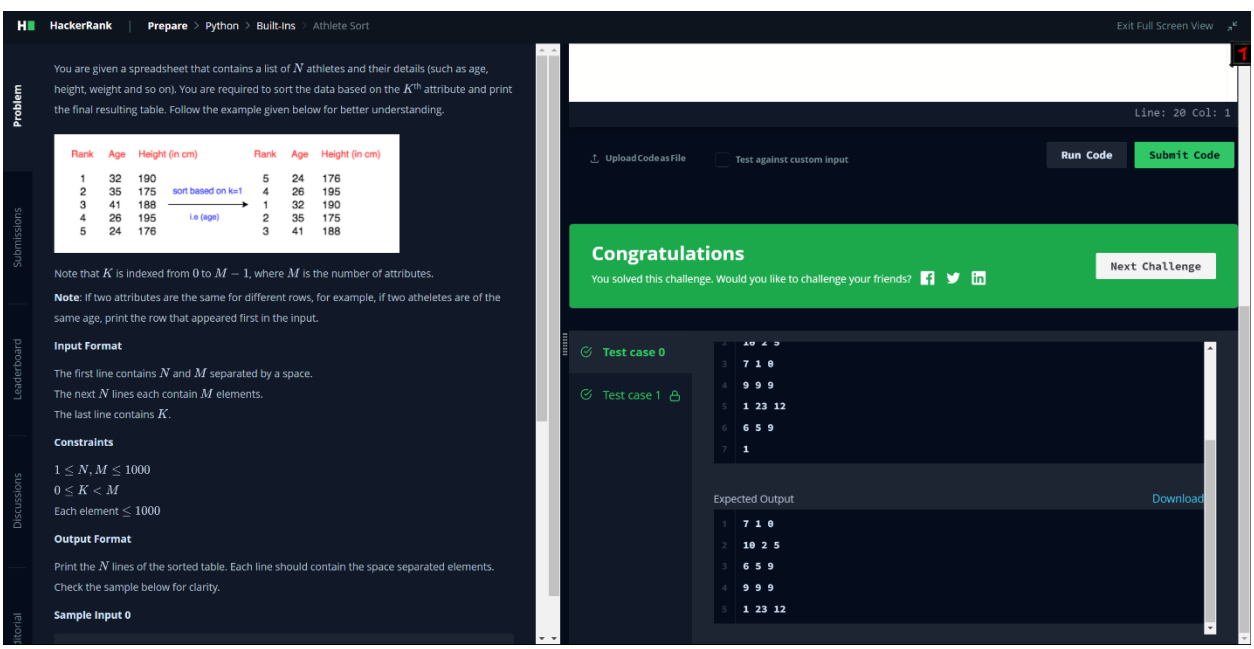
```
import math
class Complex:
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary

    def __add__(self, no):
        return Complex(self.real + no.real, self.imaginary + no.ima
ginary)

    def __sub__(self, no):
        return Complex(self.real - no.real, self.imaginary - no.ima
ginary)

    def __mul__(self, no):
        return Complex(
            self.real * no.real - self.imaginary * no.imaginary,
            self.real * no.imaginary + self.imaginary * no.real,
        )

    def __truediv__(self, no):
        divider = no.real**2 + no.imaginary**2
        return Complex(
            (self.real * no.real + self.imaginary * no.imaginary) /
 divider,
            (self.imaginary * no.real - self.real * no.imaginary) /
 divider,
        )

    def mod(self):
        return Complex(math.sqrt(self.real**2 + self.imaginary**2),
 0.00)

    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')
```



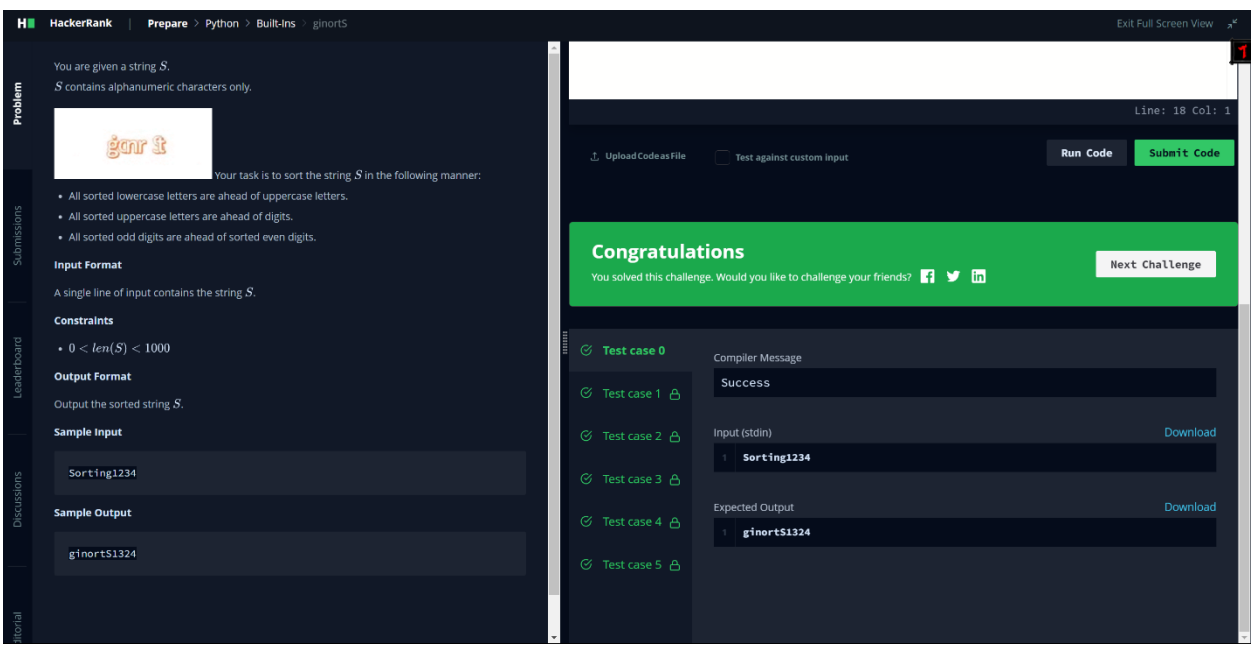
MEDIUM 14

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


if __name__ == "__main__":
    n, m = map(int, input().split())

    arr = []

    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))

    k = int(input())

    for i in sorted(arr, key=lambda x: x[k]):
        print(*i)
```



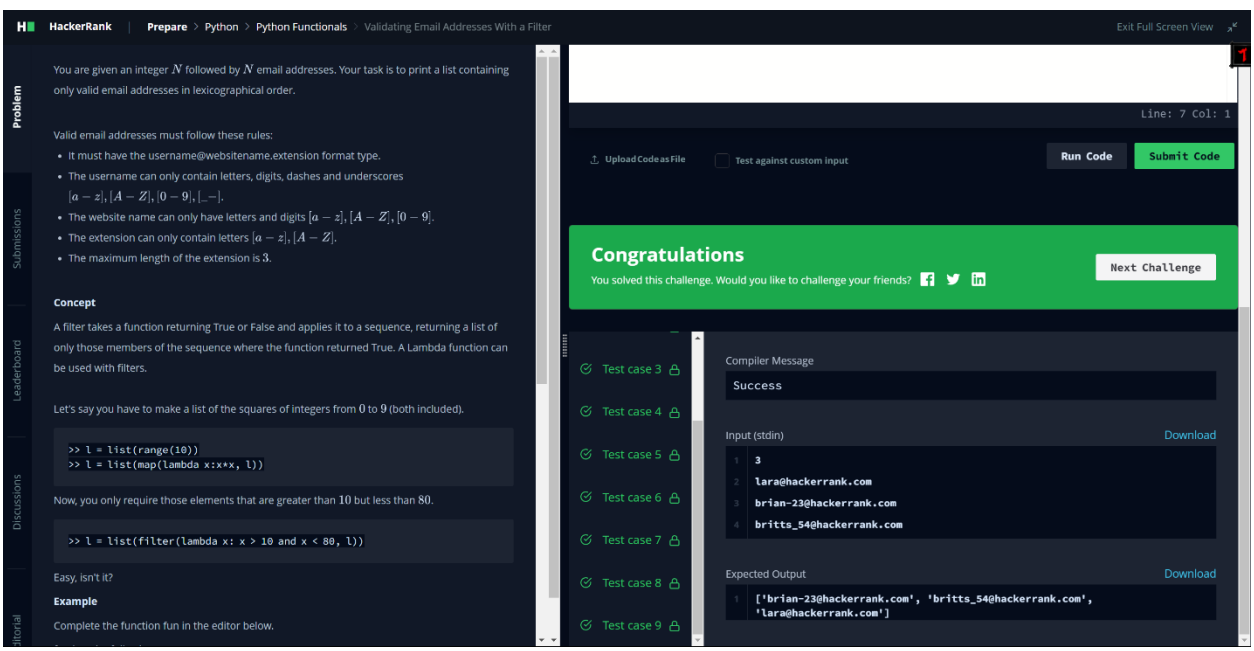
MEDIUM 15

```python
s = input()
print(
    "".join(
        sorted(
            s,
            key=lambda x: (
                x.isdigit() and int(x) % 2 == 0,
                x.isdigit(),
                x.isupper(),
                x.islower(),
                x,
            ),
        )
    )
)
```



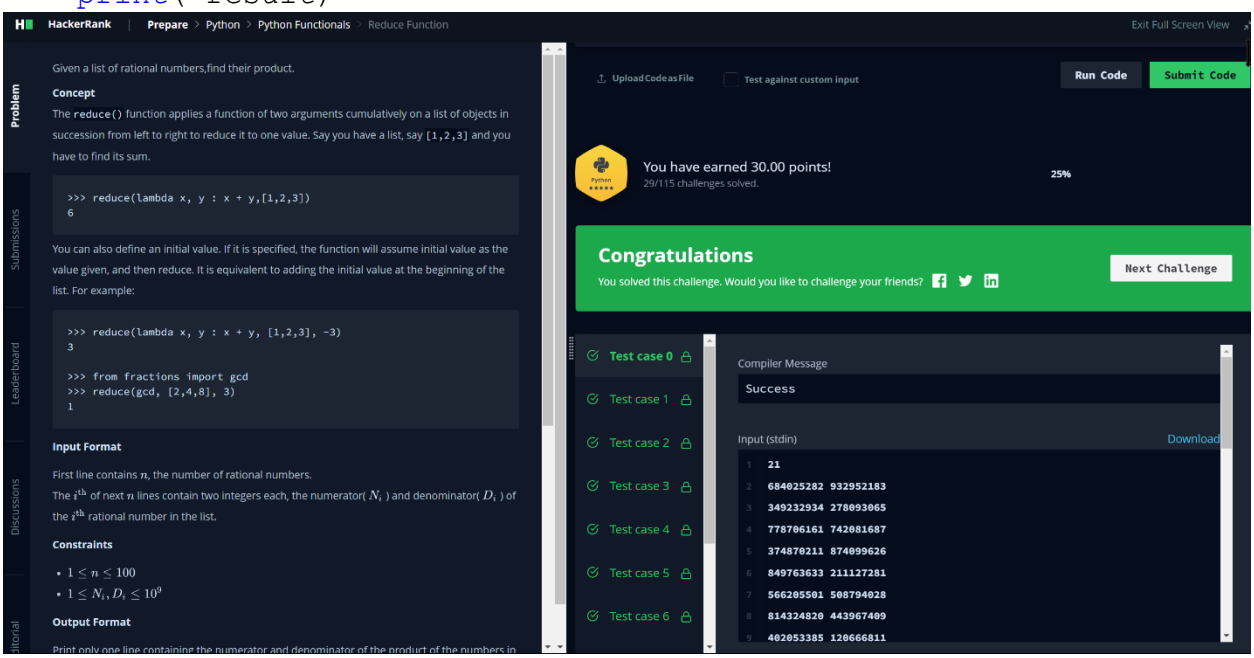
MEDIUM 16

```python
import re


def fun(s):
    return re.search(r"^[\w-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$", s)
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)
```



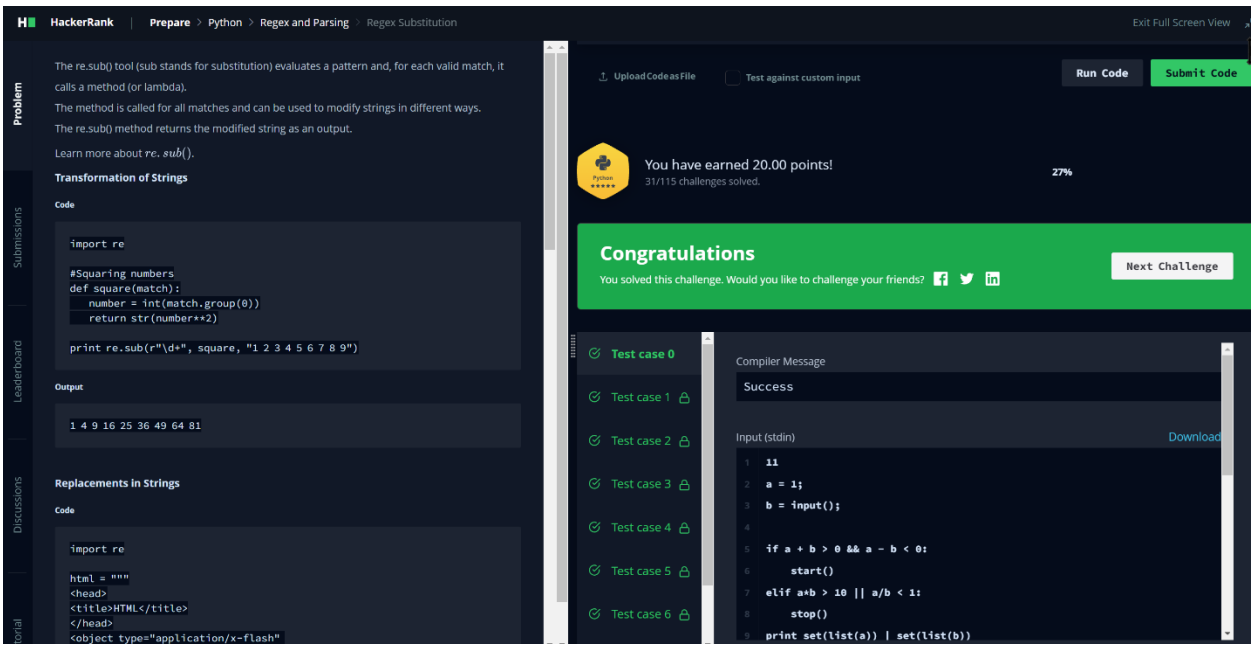
MEDIUM 17

```
om fractions import Fraction
from functools import reduce
def product(fracs):
    t = Fraction(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)
```



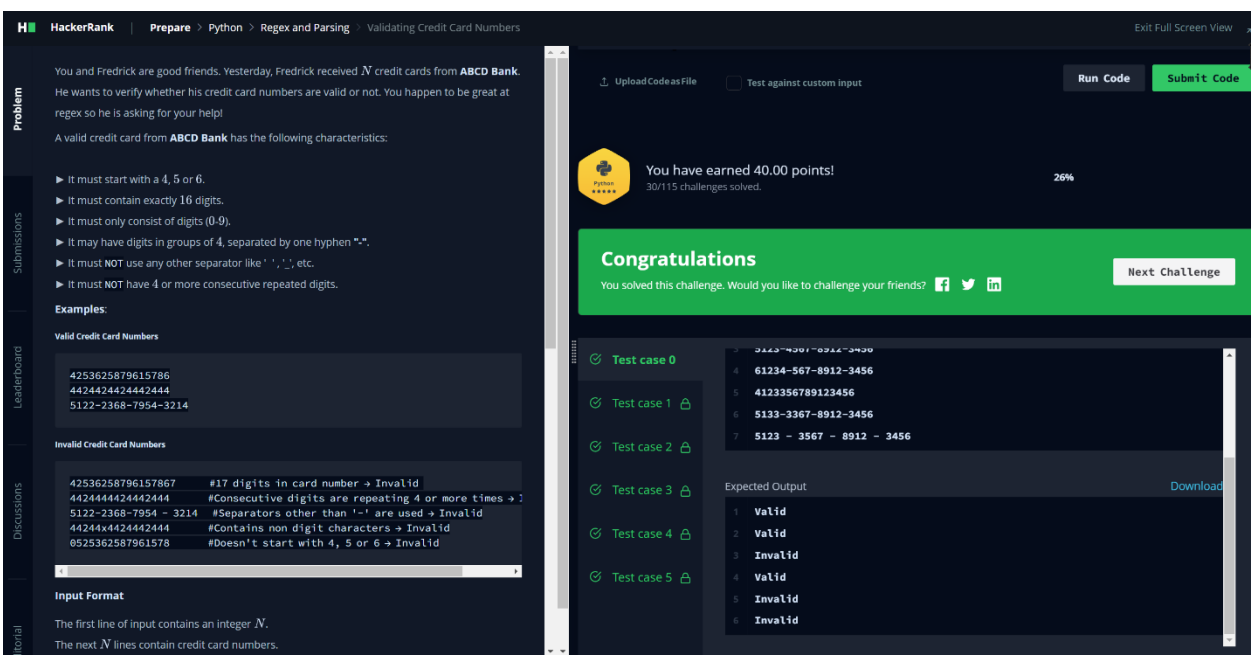
MEDIUM 18

```python
import re
import sys

n = int(input())
for line in sys.stdin:
    remove_and = re.sub(r"(?<= )(&&)(?= )", "and", line)
    remove_or = re.sub(r"(?<= )(\|\|)(?= )", "or", remove_and)
    print(remove_or, end="")
```



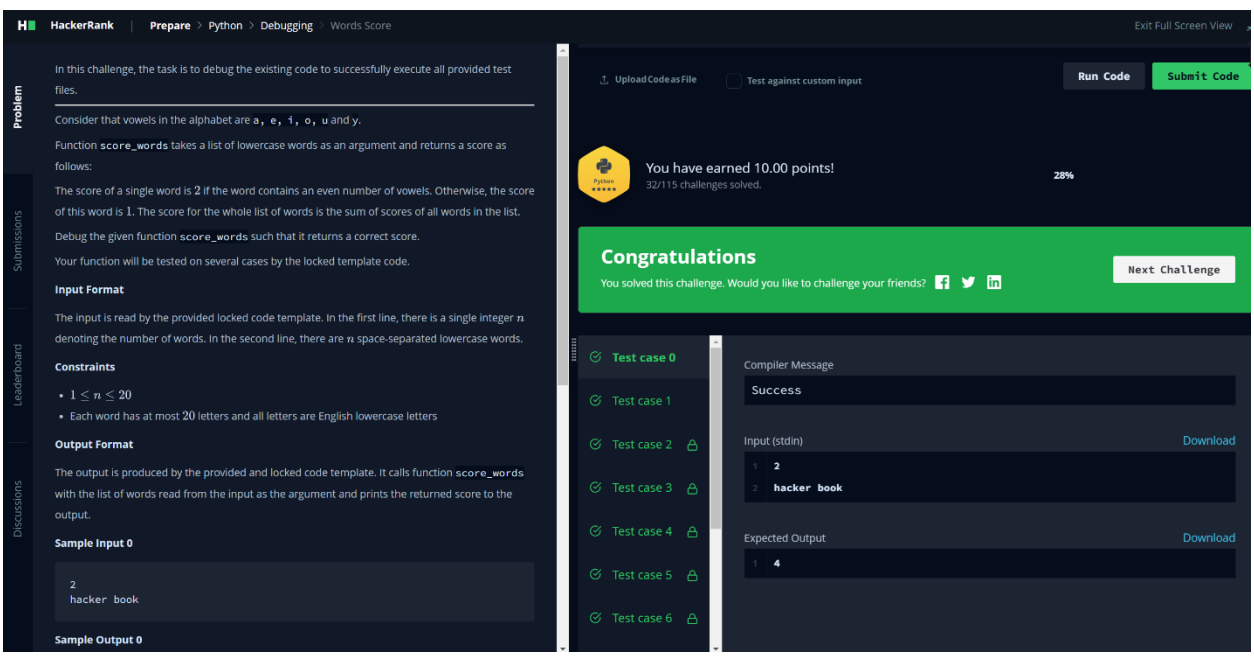
MEDIUM 19

```python
import re

n = int(input())
for _ 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:
        print("Valid")
    else:
        print("Invalid")
```



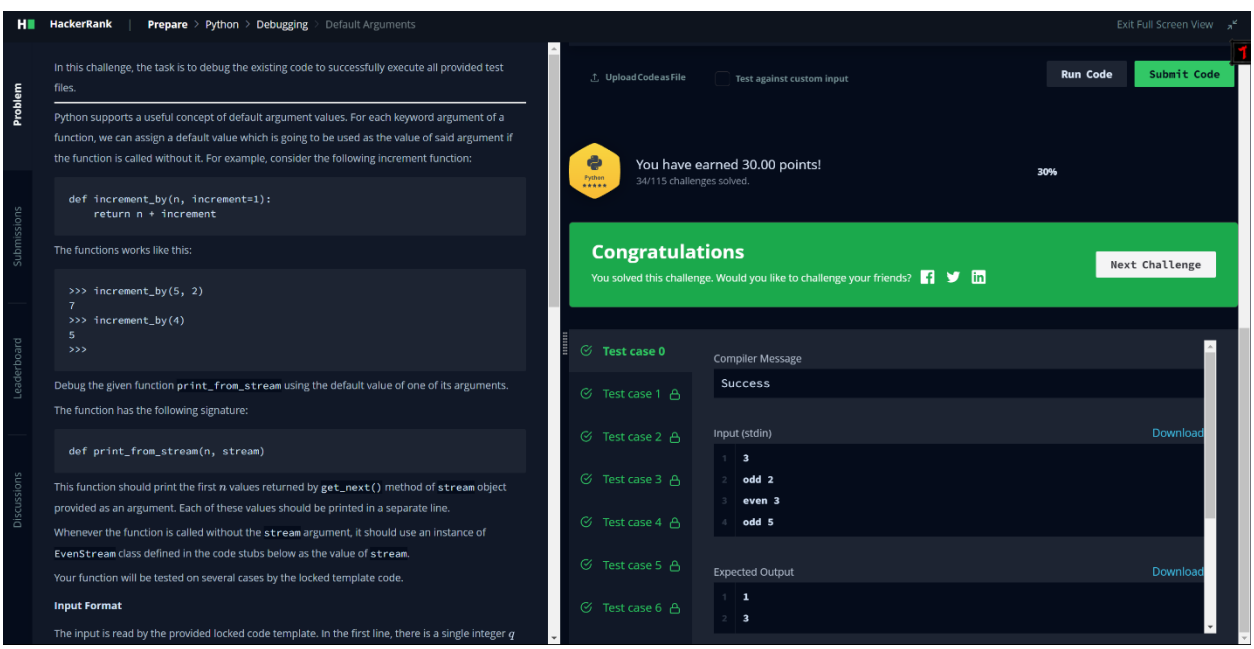
MEDIUM 20

```python
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 = sum(1 for letter in word if is_vowel(letter))
        score += 2 if num_vowels % 2 == 0 else 1
    return score


n = int(input())
words = input().split()
print(score_words(words))
```



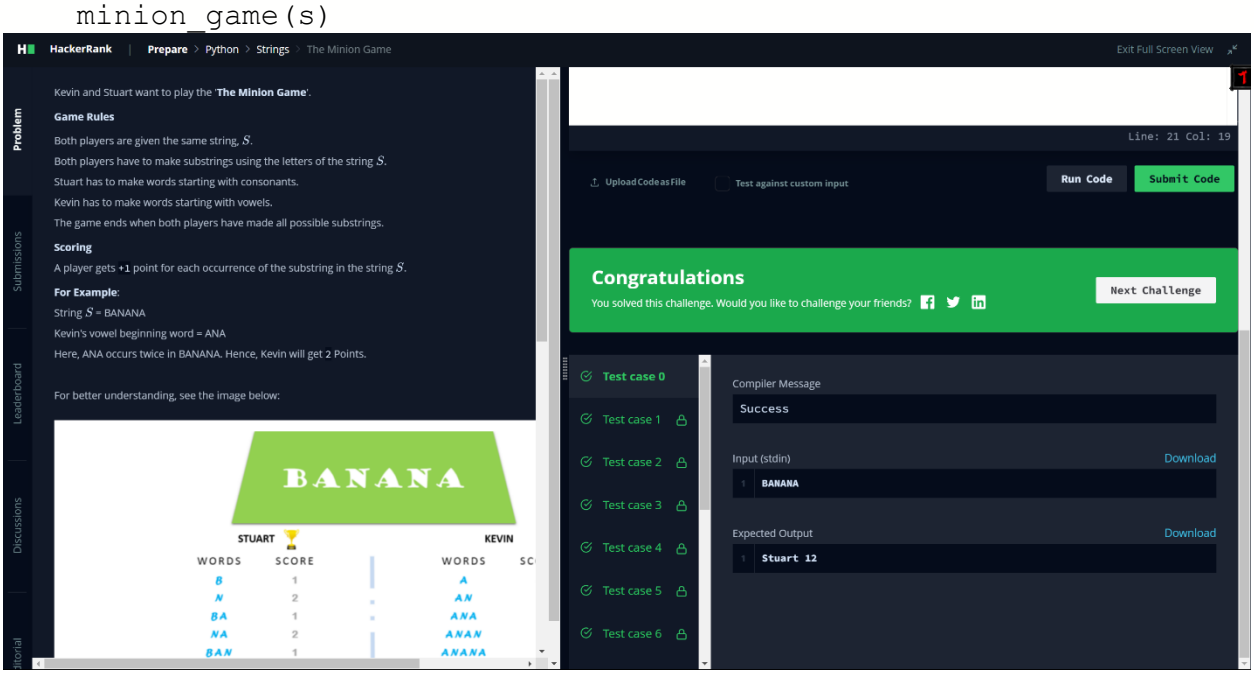
MEDIUM 21

```python
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
```



MEDIUM 22

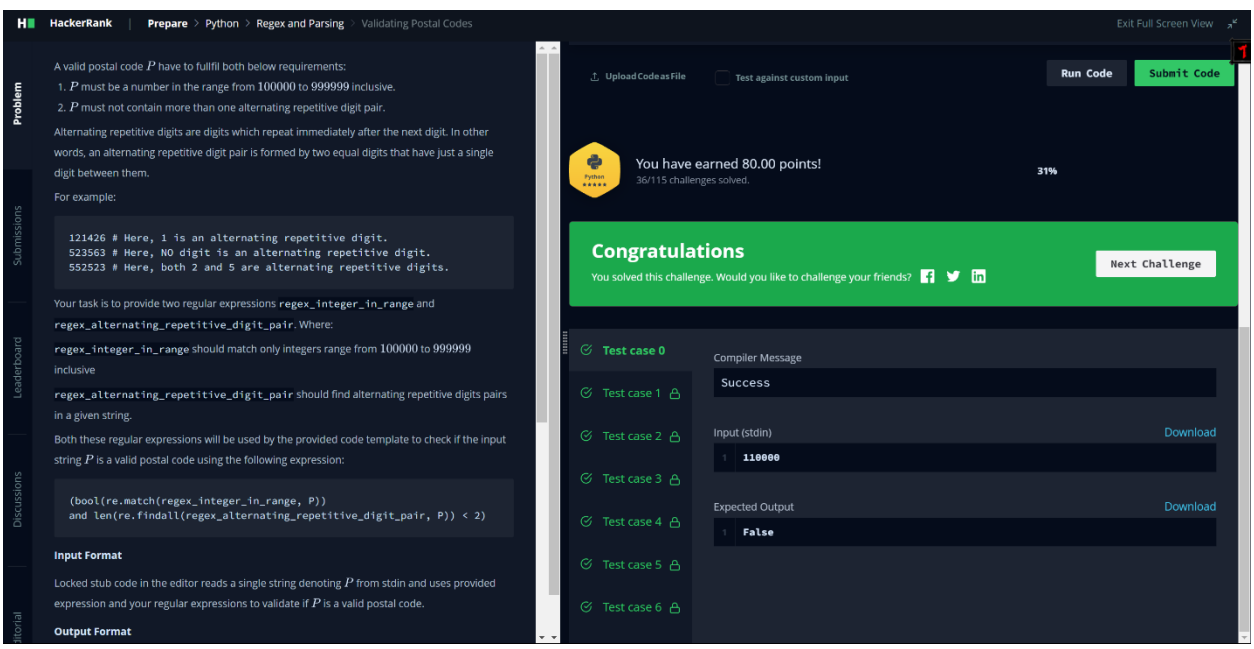
```python
def minion_game(string):

    s_length = len(s)
    vowel_list = ["A", "E", "I", "O", "U"]
    stuart_point = 0
    kevin_point = 0
    for i in range(s_length):
        if s[i] in vowel_list:
            kevin_point += s_length - i
        else:
            stuart_point += s_length - i
    if stuart_point == kevin_point:
        print("Draw")
    elif kevin_point > stuart_point:
        print("Kevin", kevin_point)
    else:
        print("Stuart", stuart_point)

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



HARD 01

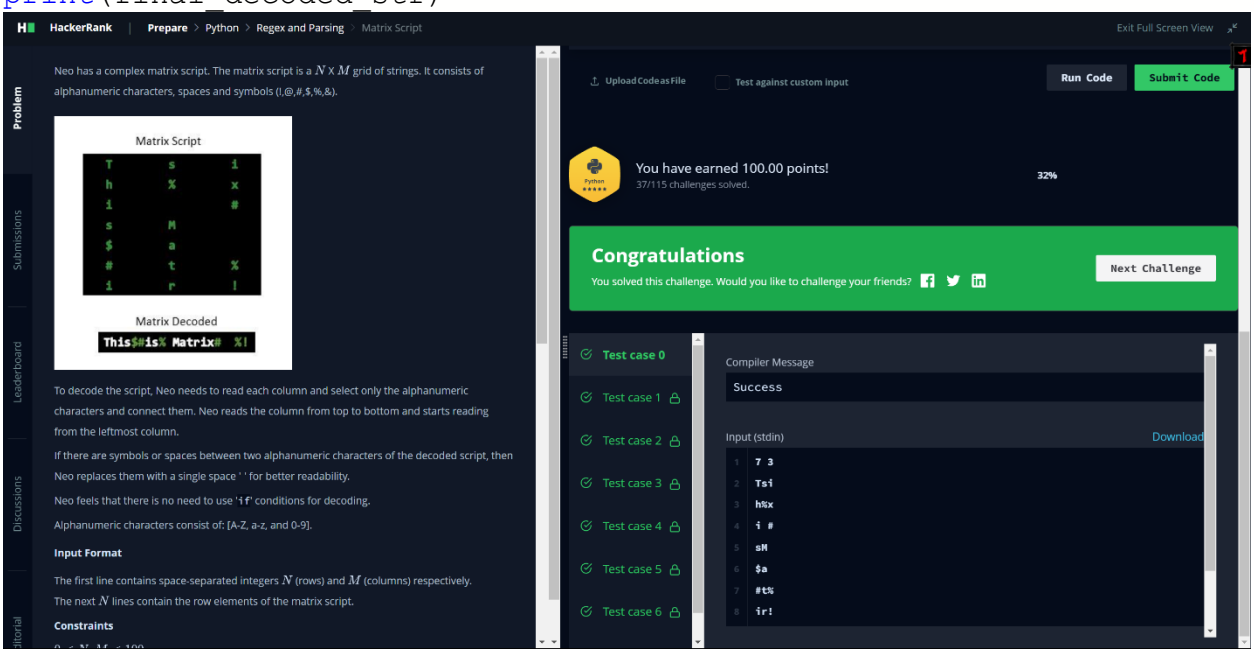
```python
import re

p = input().strip()
range_check = bool(re.match(r"^[1-9][0-9]{5}$", p))
repeat_check = len(re.findall(r"(?=([0-9])[0-9]\1)", p))
print(range_check == True and repeat_check < 2)
```

```python
import re
P = input()

print (bool(re.match(regex_integer_in_range, P))
and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2
)
```



HARD 02

```python
import re

n, m = map(int, input().split())
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)
```



HARD 03

```python
import itertools

k, m = map(int, input().split())

main_ar = []
for _ in range(k):
    ar = list(map(int, input().split()))
    main_ar.append(ar[1:])

all_combination = itertools.product(*main_ar)
result = 0
for single_combination in all_combination:
    result = max(sum(x * x for x in single_combination) % m, result
)
print(result)
```