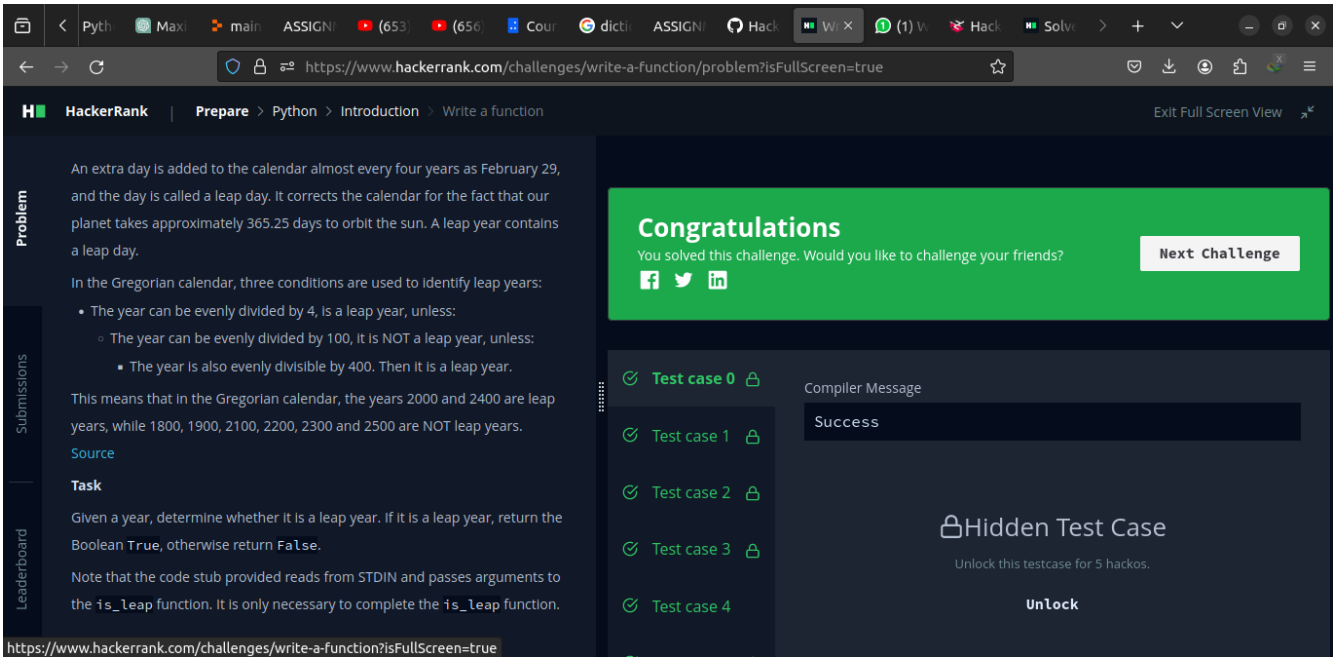
Name: Muhammad Yousuf Eisa
Roll no: 452506
Assignment no:3
Artificial Intelligence

Medium Level Challenges

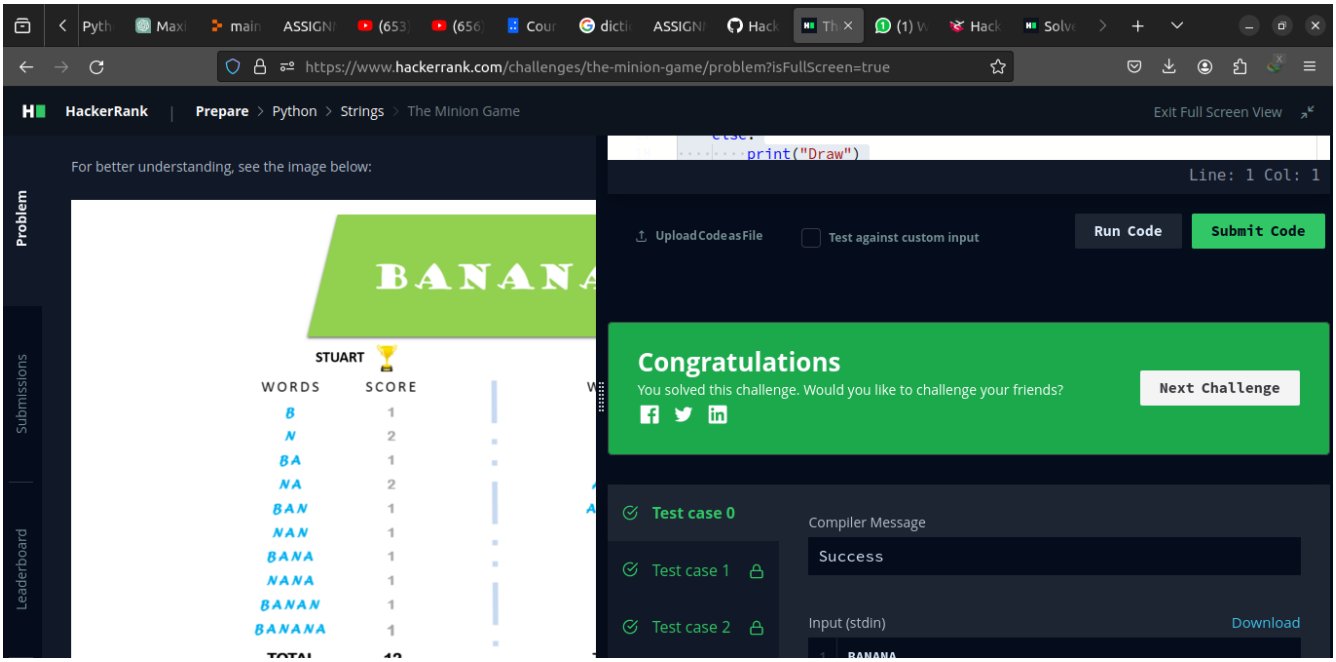
1- Write a function



Code

```
def is_leap(year):
return (year % 400 == 0) or ((year % 4 == 0) and (year % 100 != 0))
year = int(input())
print(is_leap(year))
```

2- Minion Game

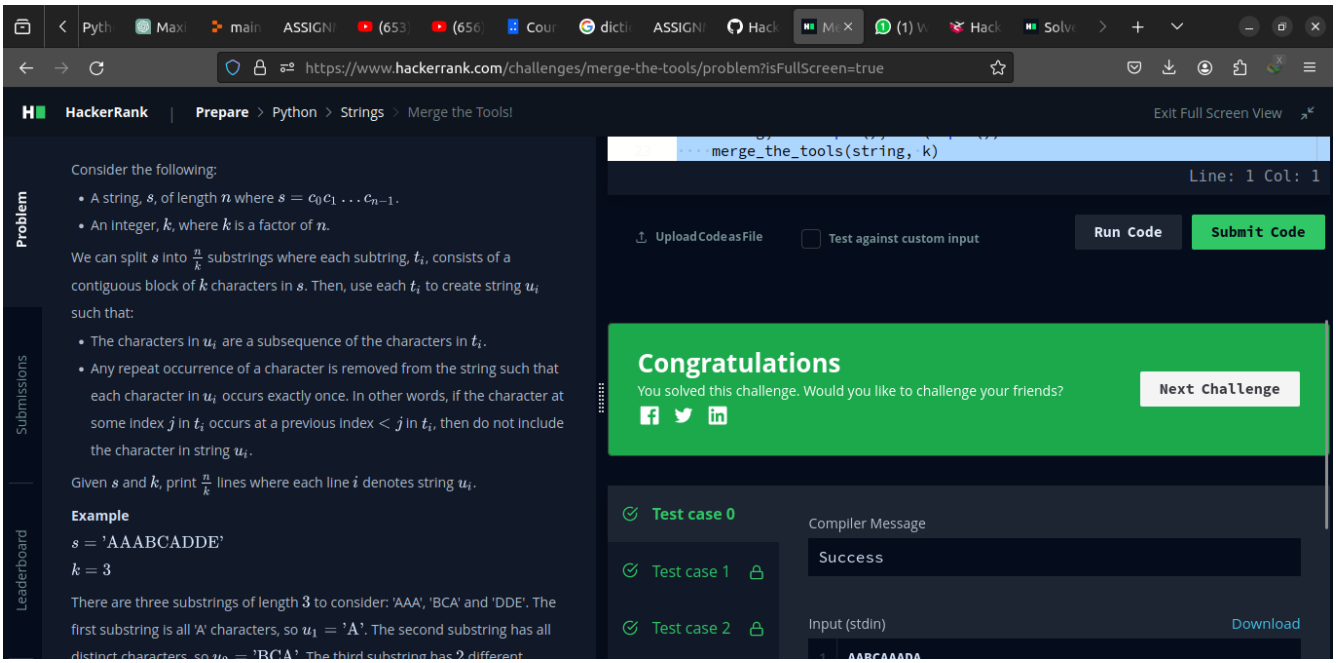


Code

vowels = ['A', 'E', 'I', 'O', 'U']

```
def minion_game(string):
score_kevin = 0
score_stuart = 0
for ind in range(len(string)):
if string[ind] in vowels:
score_kevin += len(string) - ind
else:
score_stuart += len(string) - ind
if score_kevin > score_stuart:
print("Kevin {}".format(score_kevin))
elif score_kevin < score_stuart:
print("Stuart {}".format(score_stuart))
else:
print("Draw")
if __name__ == '__main__':
s = input()
minion_game(s)
```

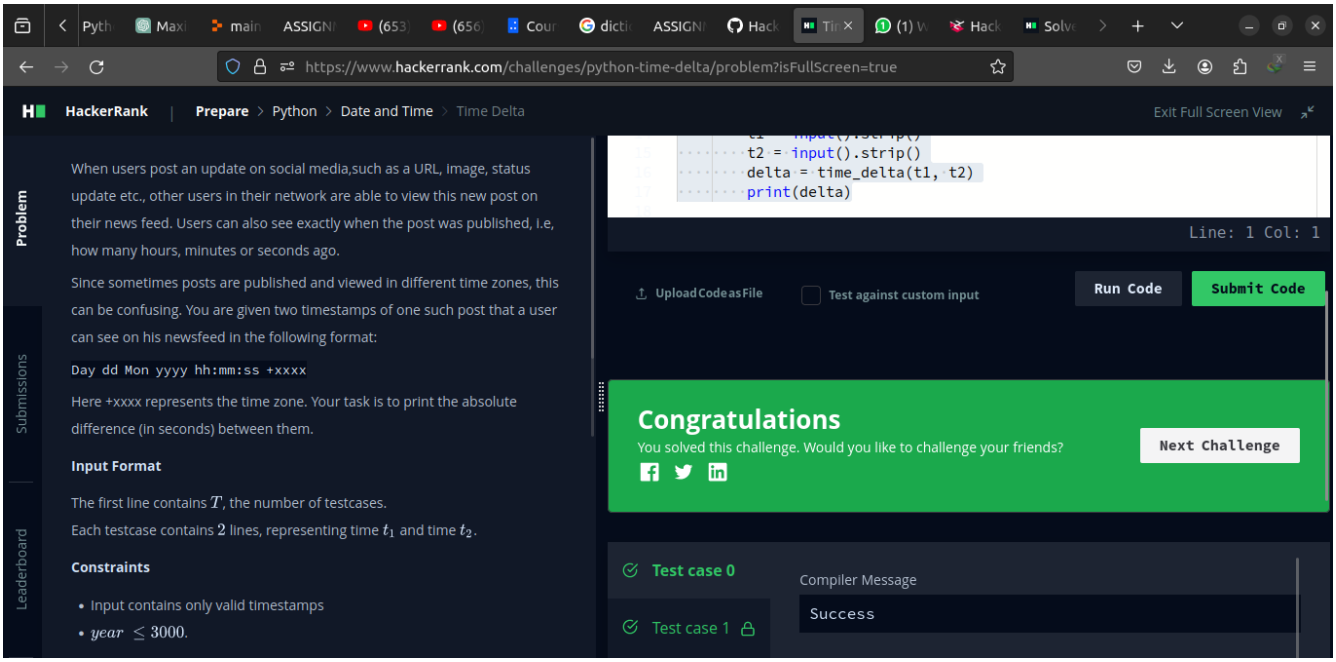
3- Merge the tools



Code

```
def merge_the_tools(string, k):
block_cnt = len(string)//k
output_t = []
output_u = []
#print("{}/{} = {}".format(len(string), k, block_len))
for ind in range(0, len(string) - k + 1, k):
output_t.append(string[ind:ind + k])
for block in output_t:
for char in block:
char_count = block.count(char)
if char_count > 1:
block = block[::-1]
block = block.replace(char, '', char_count - 1)
block = block[::-1]
output_u.append(block)
print("\n".join(map(str, output_u)))

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

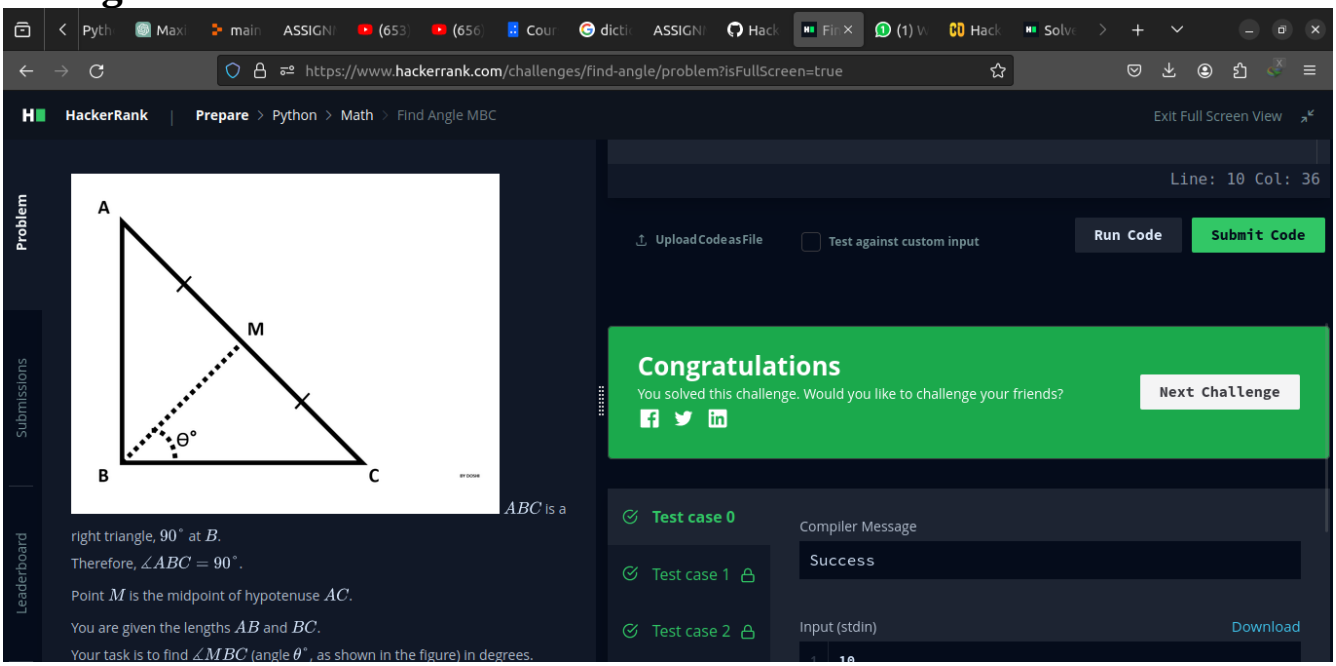
4- Time Delta



Code

```
import sys
from datetime import datetime as dt

dformat = "%a %d %b %Y %H:%M:%S %z"
def time_delta(t1, t2):
first = dt.strptime(t1, dformat)
second = dt.strptime(t2, dformat)
return int(abs((first - second).total_seconds()))

if __name__ == "__main__":
t = int(input().strip())
for a0 in range(t):
t1 = input().strip()
t2 = input().strip()
delta = time_delta(t1, t2)
print(delta)
```

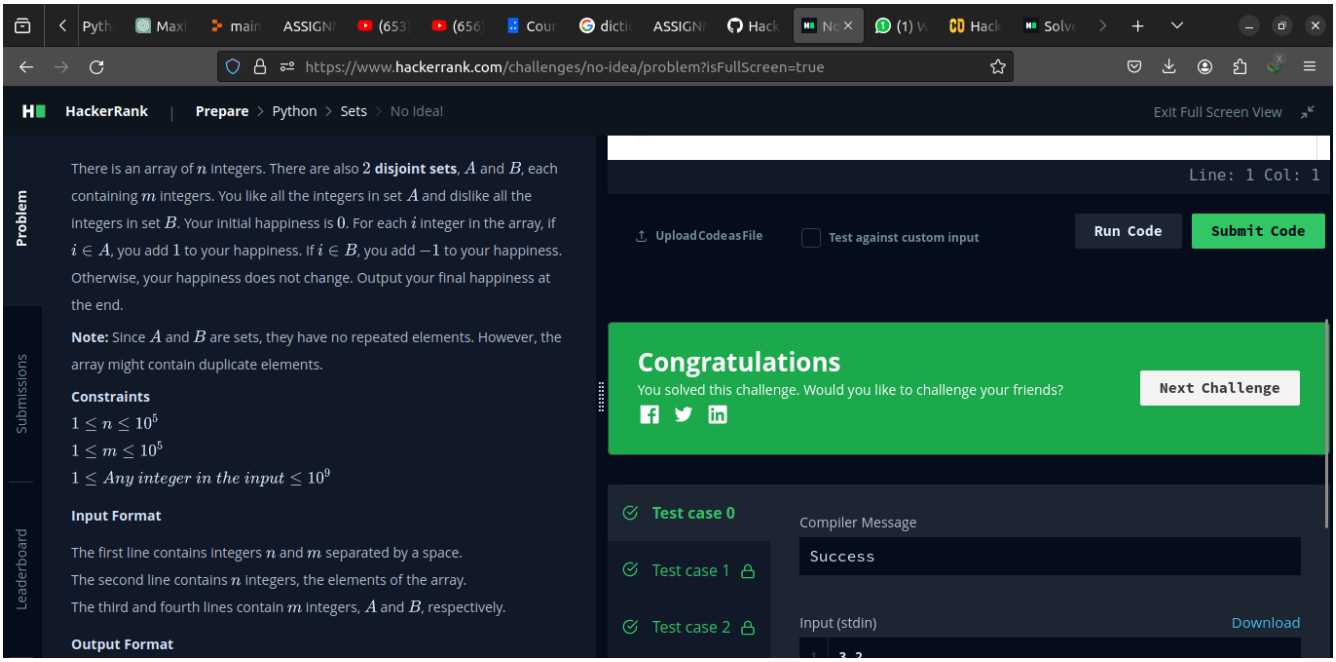
5- Angle MBC



Code

```
from math import atan
from math import degrees

if __name__ == "__main__":
ab = int(input().strip())
bc = int(input().strip())
print("{}°".format(int(round(degrees(atan(ab/bc))))))
```

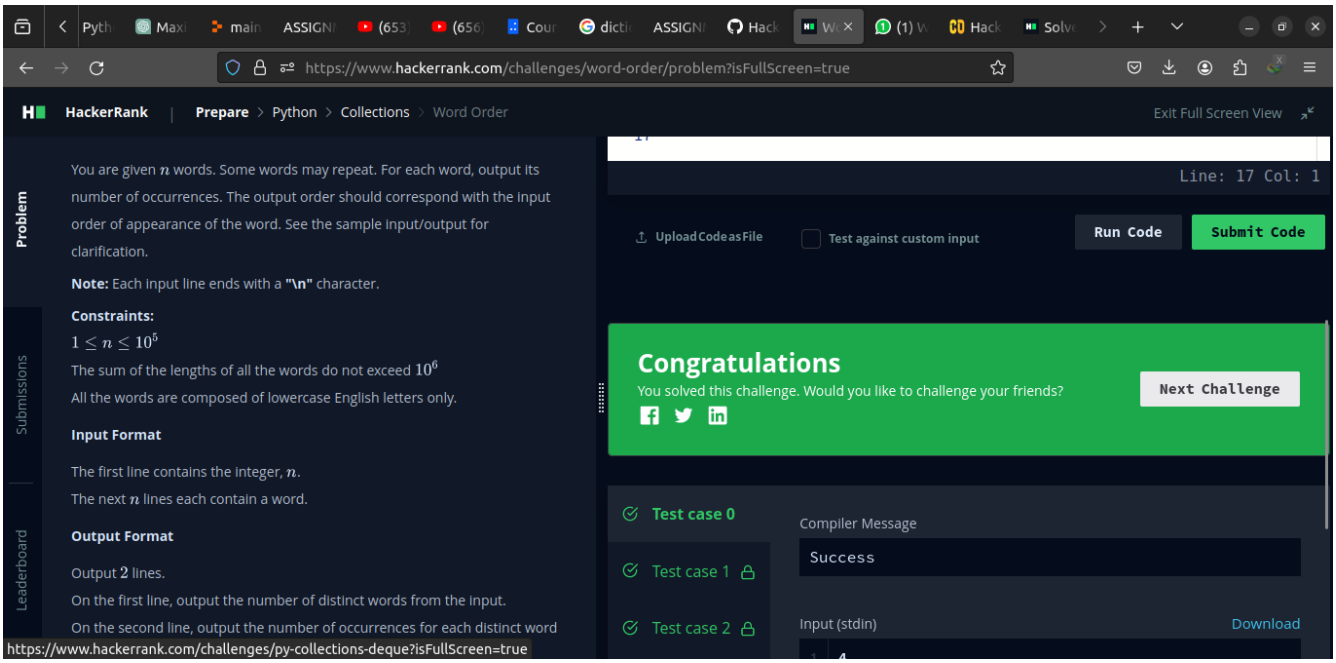
6- No Ideal



Code

```
if __name__ == "__main__":
happiness = 0
n, m = map(int, input().strip().split(' '))
arr = list(map(int, input().strip().split(' ')))
good = set(map(int, input().strip().split(' ')))
bad = set(map(int, input().strip().split(' ')))
for el in arr:
if el in good:
happiness += 1
elif el in bad:
happiness -= 1
print(happiness)
```

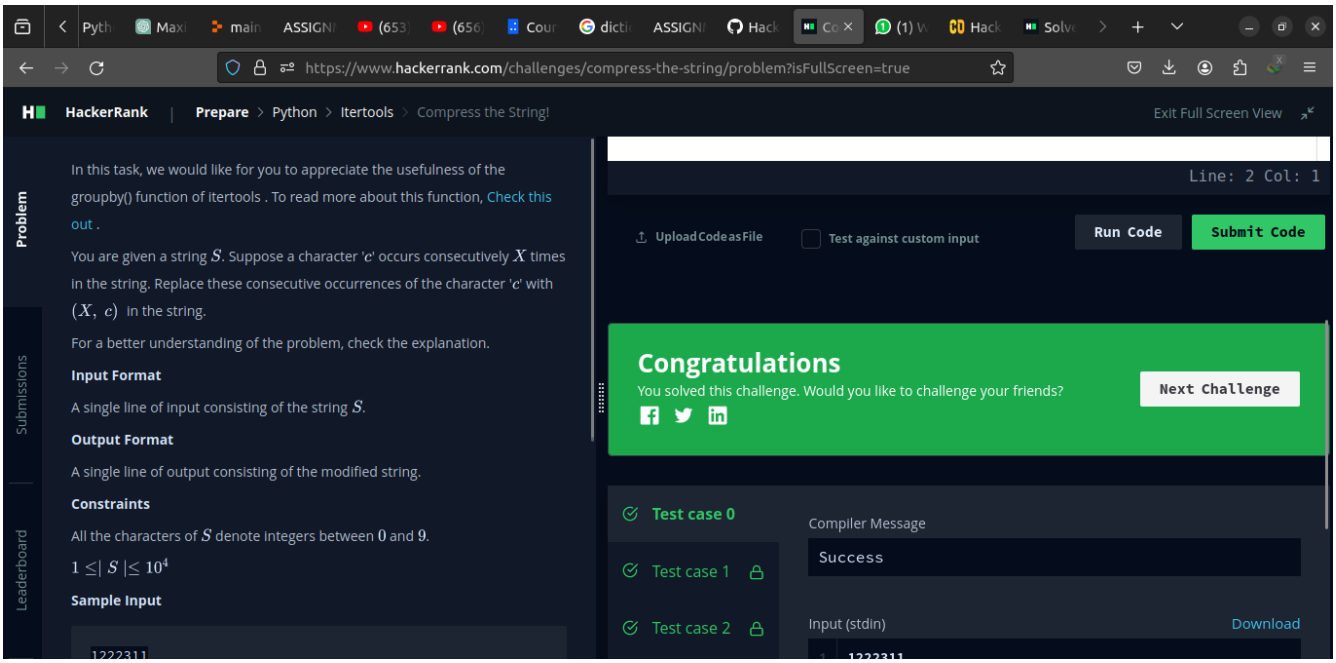
7- Word Order



Code

```
from collections import OrderedDict

if __name__ == "__main__":
num = int(input().strip())
history = OrderedDict()
for _ in range(num):
word = str(input().strip().split())
if word not in history.keys():
history[word] = 1
else:
history[word] += 1
print(len(history.keys()))
print(" ".join(map(str, history.values())))
```

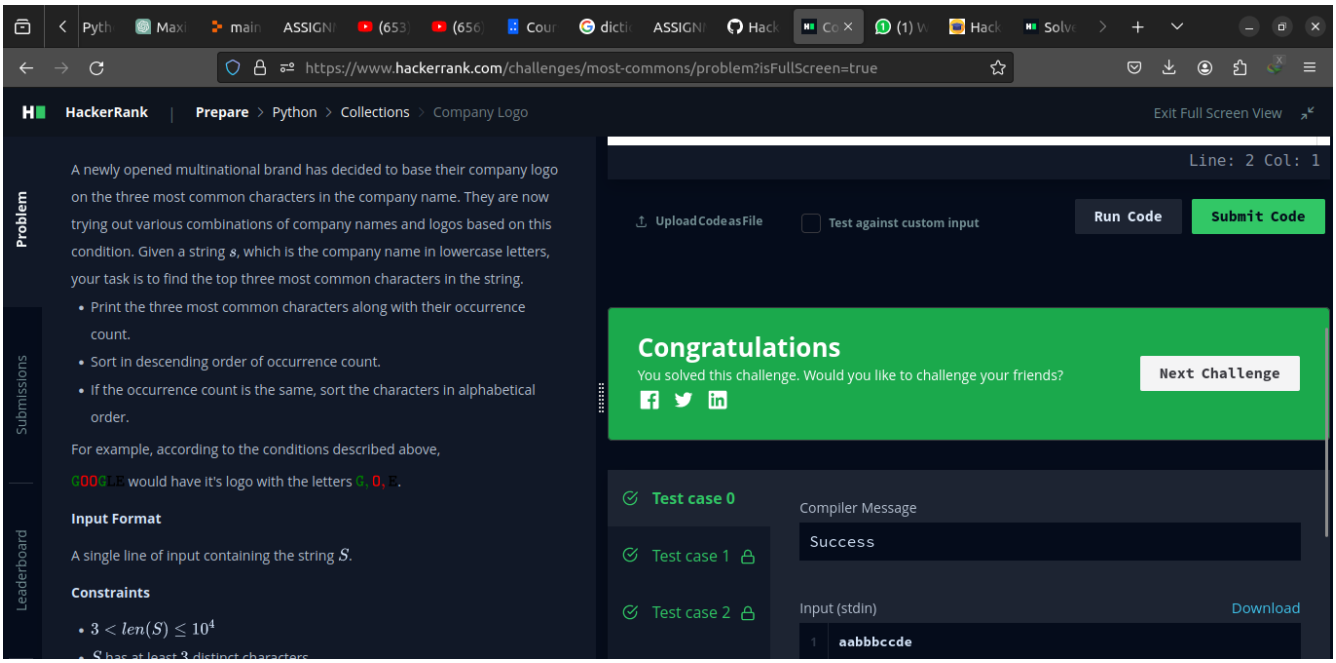
8- Compress the string



Code

```
from itertools import groupby

if __name__ == "__main__":
#in_data = input().strip().split(' ')
for el, el_list in groupby(input()):
print((len(list(el_list)), int(el)), end=' ')
```

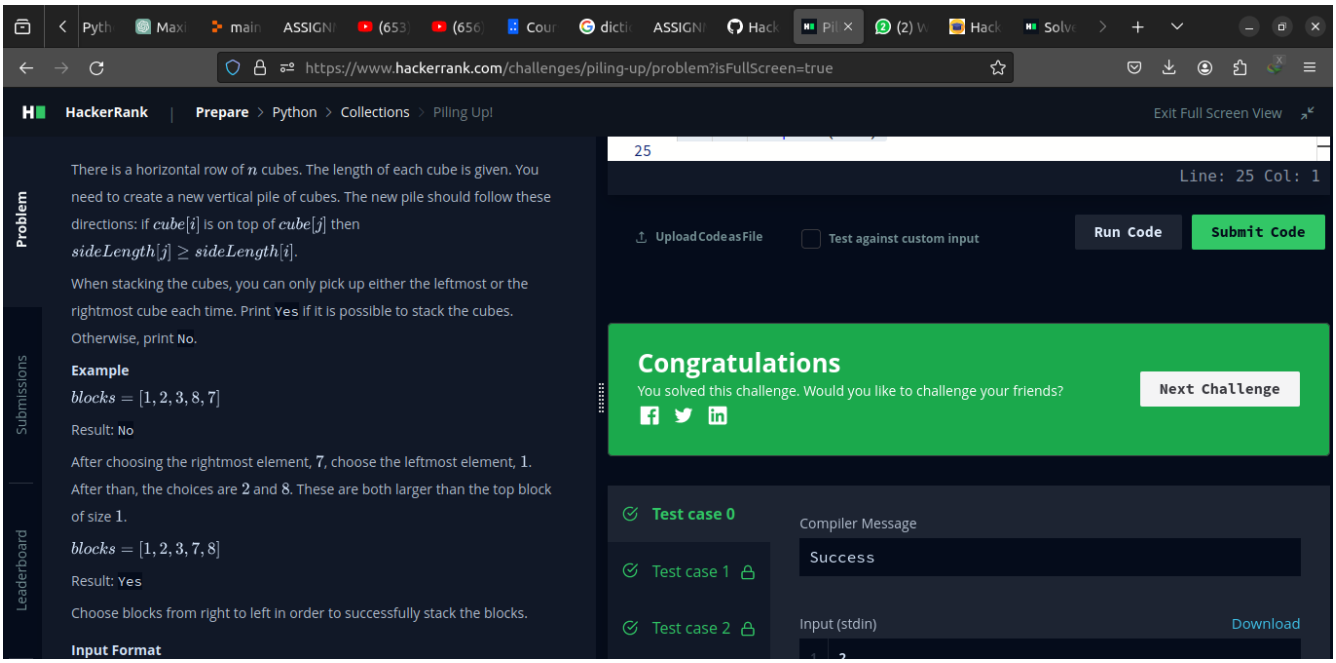
9- Company logo



Code

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

from collections import Counter, OrderedDict

class OrderedCounter(Counter, OrderedDict):
pass
[print(*c) for c in OrderedCounter(sorted(input())).most_common(3)]
```

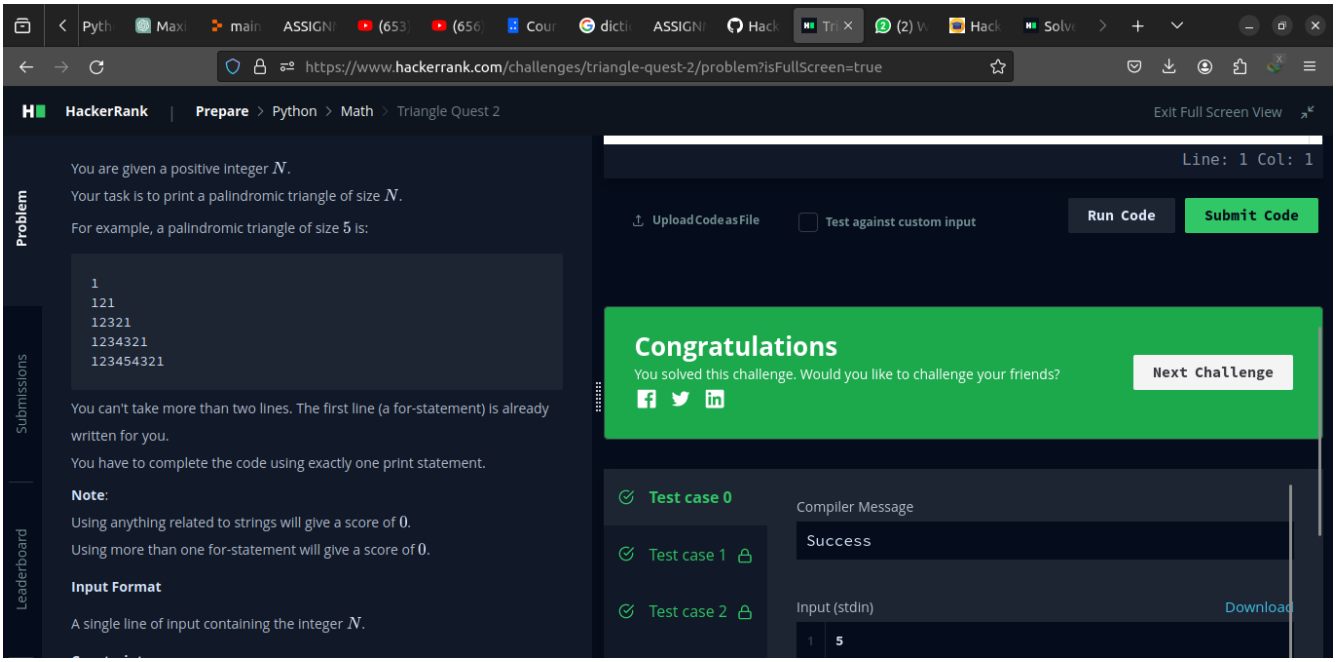
10- Pilling up



Code

```
from collections import deque

if __name__ == "__main__":
t = int(input().strip())
for _ in range(t):
num_cnt = int(input().strip())
deq = deque(list(map(int, input().strip().split(' '))))
prev = max(deq[0], deq[-1])
while deq:
if prev >= deq[0] and prev >= deq[-1]:
if deq[0] >= deq[-1]:
prev = deq.popleft()
else:
prev = deq.pop()
else:
break
if len(deq) == 0:
print('Yes')
else:
print('No')
```

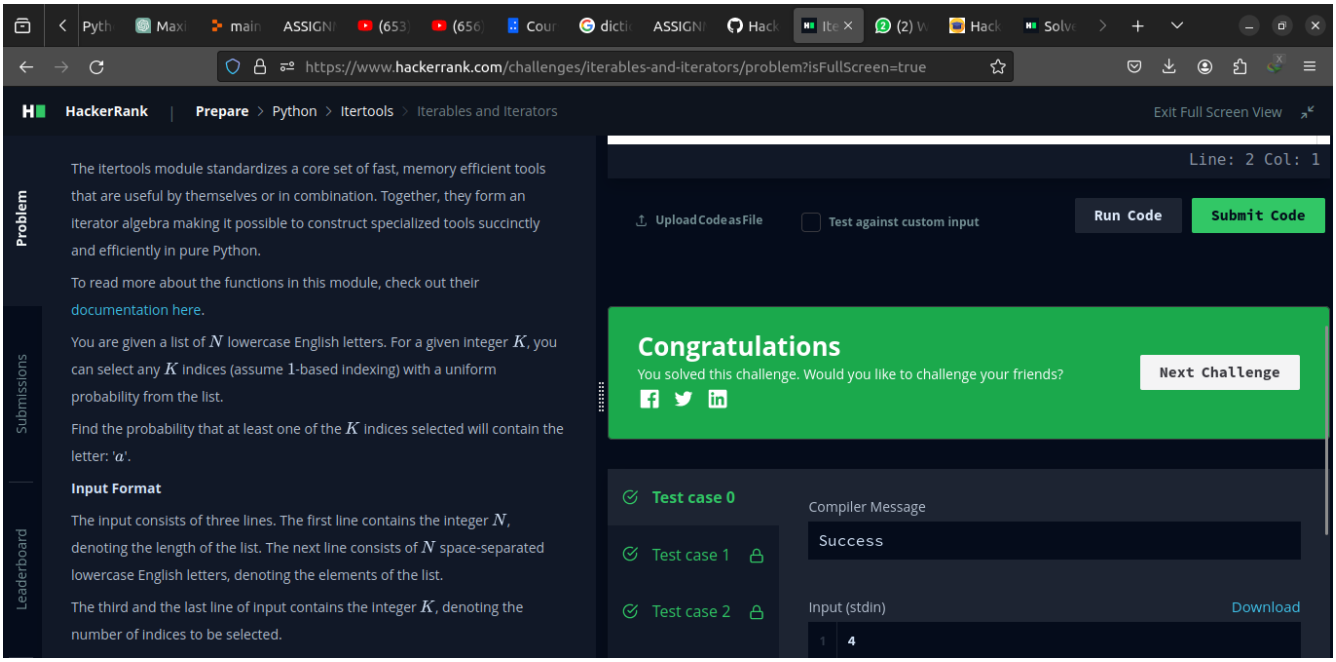
11- Triangle Quest 2



Code

```
for i in range(1,int(input())+1):
print((10**i//9)**2)
```

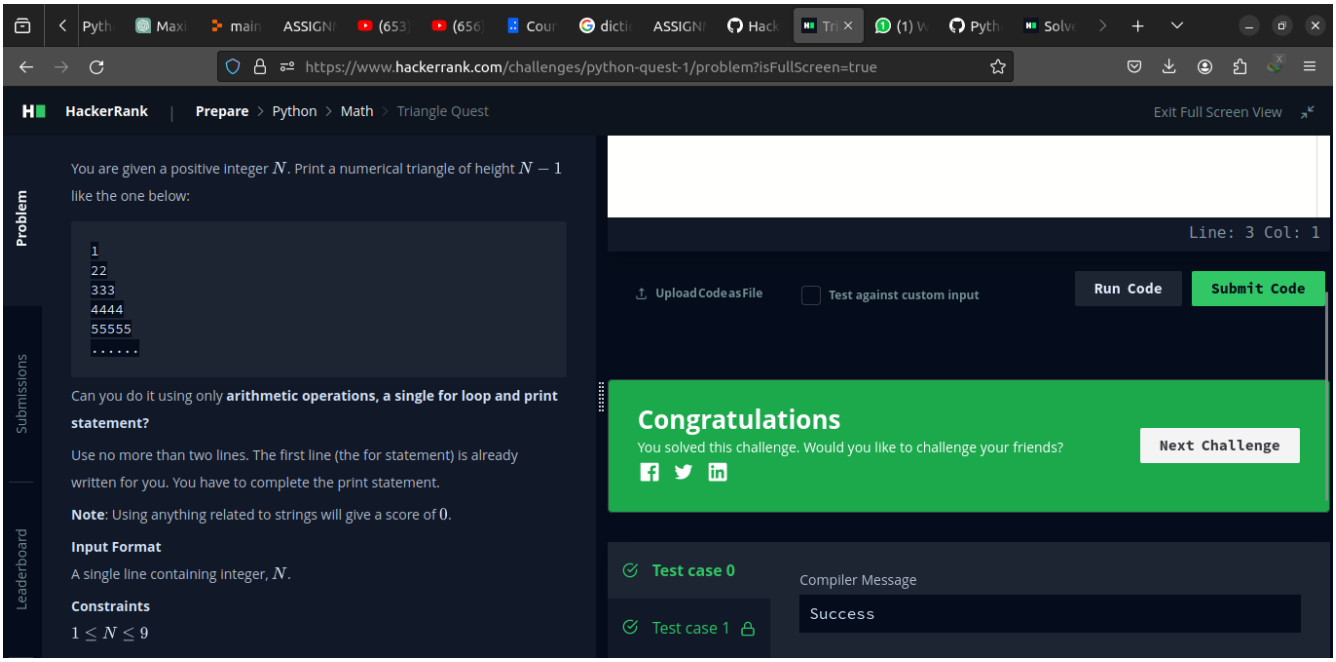
12- Iterables and Itearators



Code

```
import string
symbols = string.ascii_lowercase

from itertools import combinations

if __name__ == "__main__":
n = int(input().strip())
arr = list(map(str, input().strip().split(' ')))
times = int(input().strip())
cmbts = list(combinations(sorted(arr), times))
print("{:.4f}".format(len(list(filter(lambda a: a[0] == 'a', cmbts))))/(len(cmbts))))
```

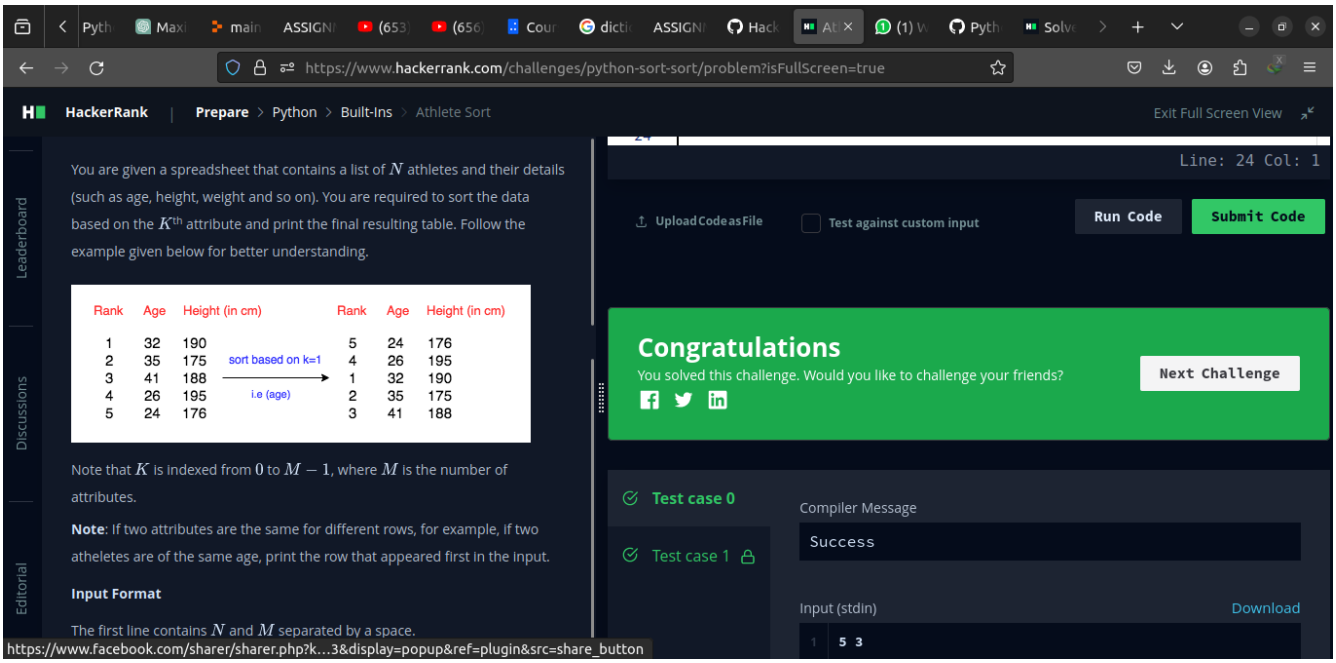
13- Triangle-Quest-1



Code

```
for i in range(1,int(input())):
print( int((i*(pow(10, i) - 1)) / 9 ))
```

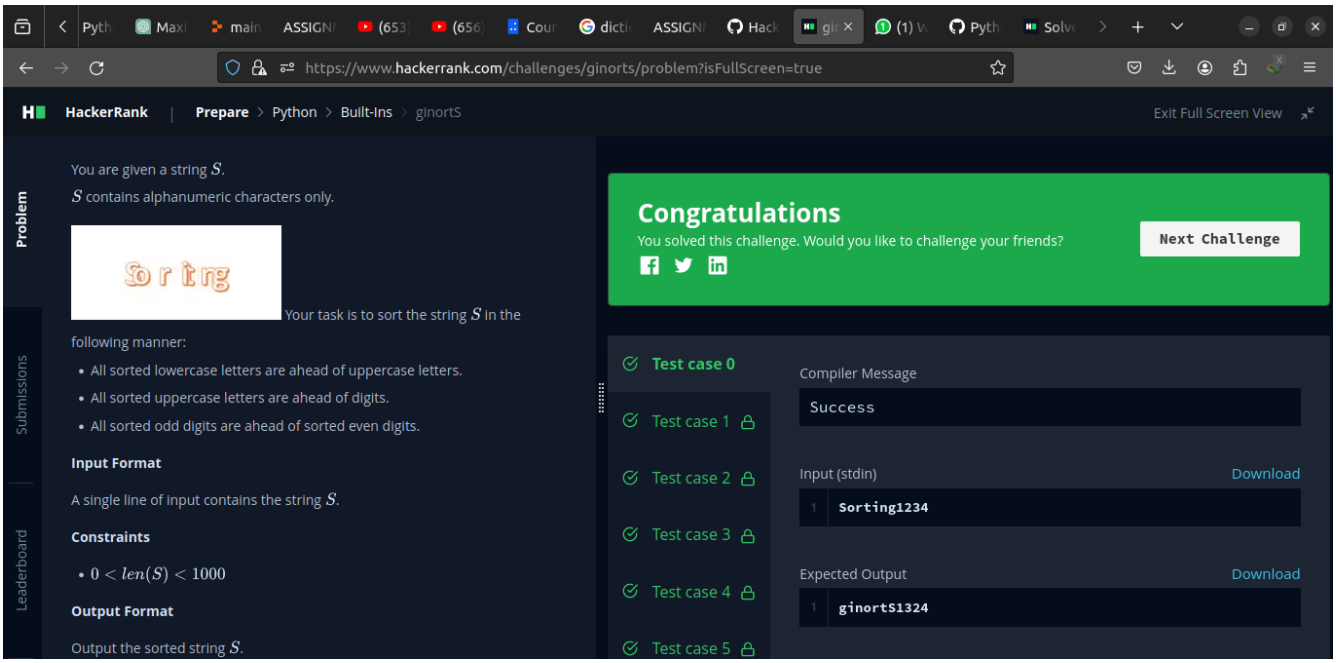
14- Athlete Sort



Code

```
import math
import os
import random
import re
import sys
if __name__ == "__main__":
n, m = input().strip().split(' ')
n, m = [int(n), int(m)]
arr = []
for arr_i in range(n):
arr_t = [int(arr_temp) for arr_temp in input().strip().split(' ')]
arr.append(arr_t)
k = int(input().strip())
for el in sorted(arr, key = lambda x: x[k]):
print(" ".join(map(str, el)))
```

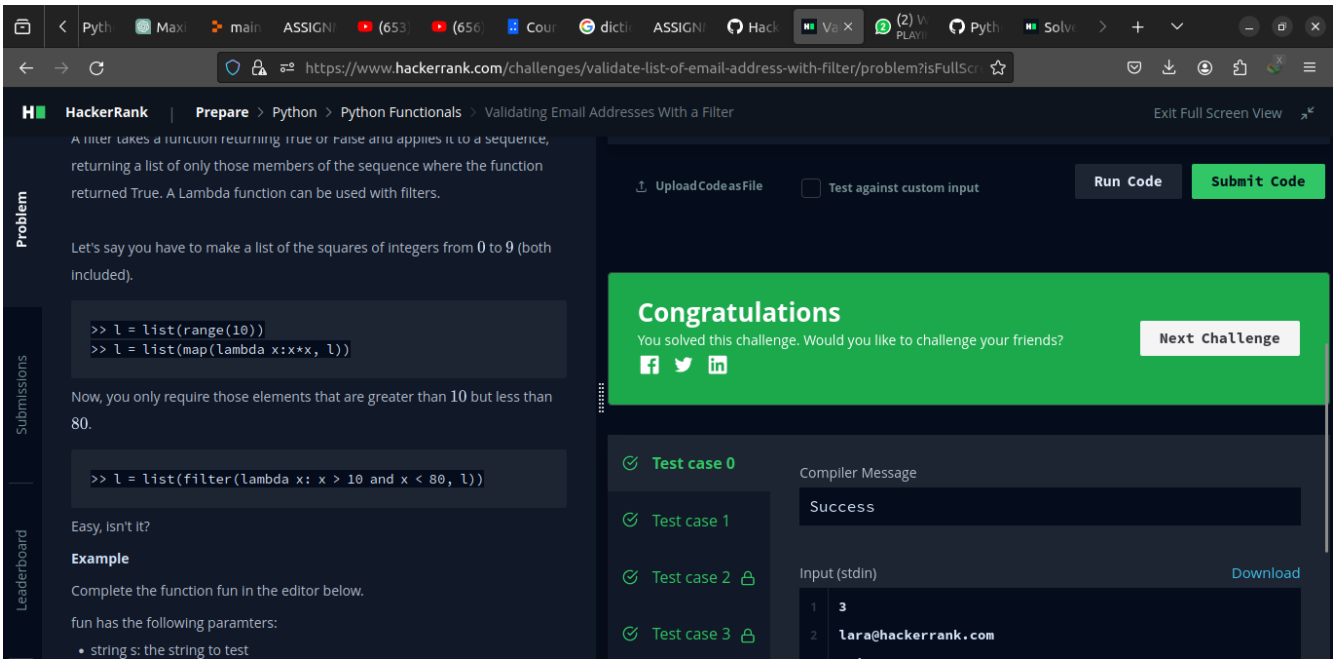
15- ginorts



Code

```
if __name__ == "__main__":
string = input().strip()
print(*sorted(string, key = lambda x: (-x.islower(), x.isdigit() - x.isupper(), x in '02468', x)), sep='')
```

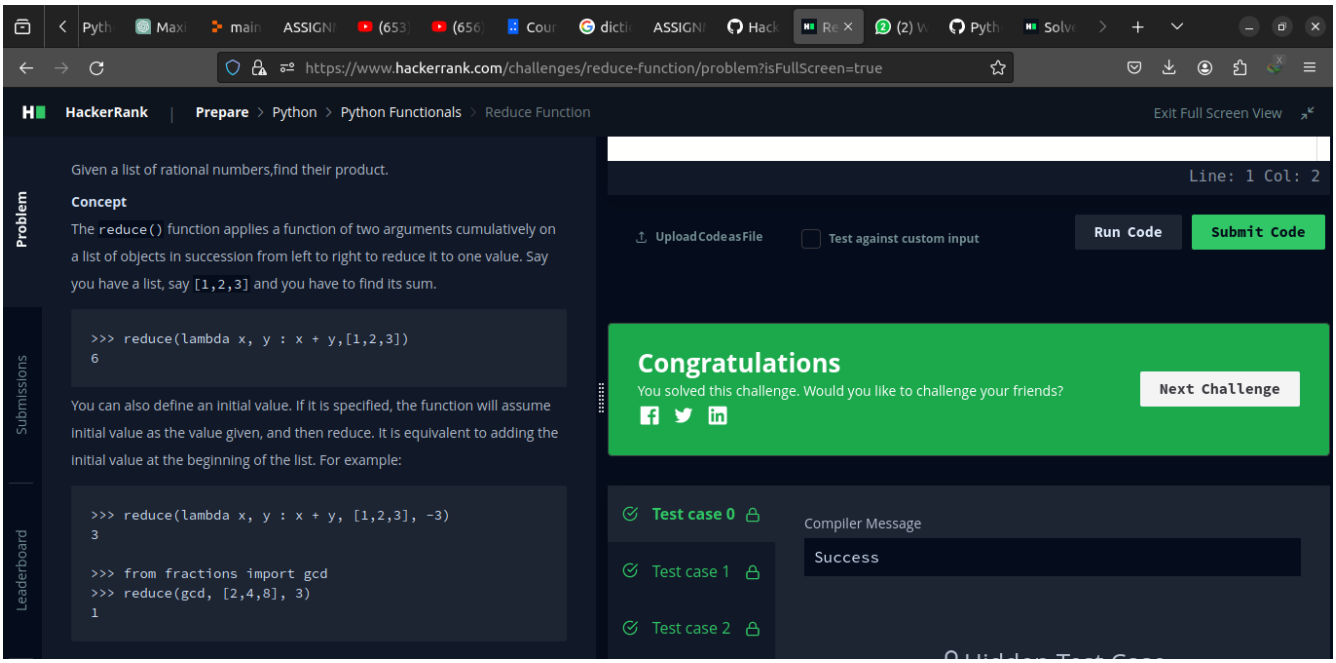
16- Validating Email address with a filter



Code

```
import re
def fun(email):
# return True if s is a valid email, else return False
#pattern = '[^@]+@[^@]+\.[^@]{1,3}'
pattern = '^[a-zA-Z][\w-]*@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'
return re.match(pattern, email)
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)
```

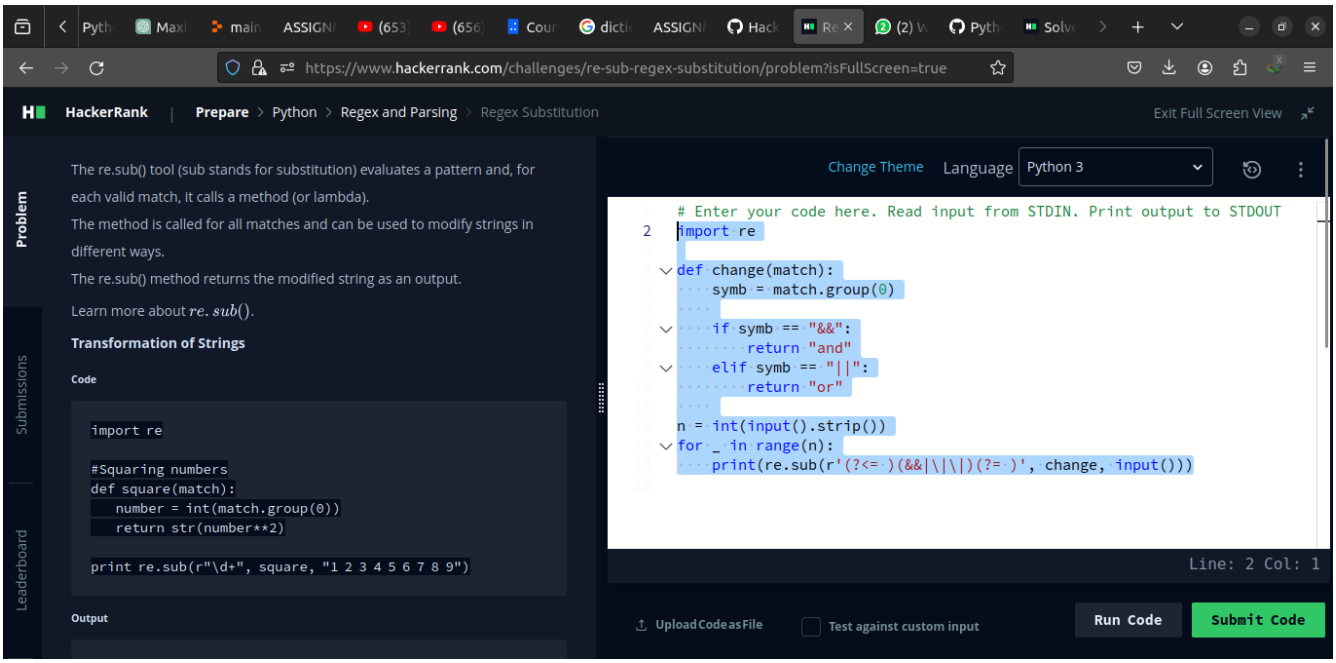
17- Reduce Function



Code

```
rom 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)
```

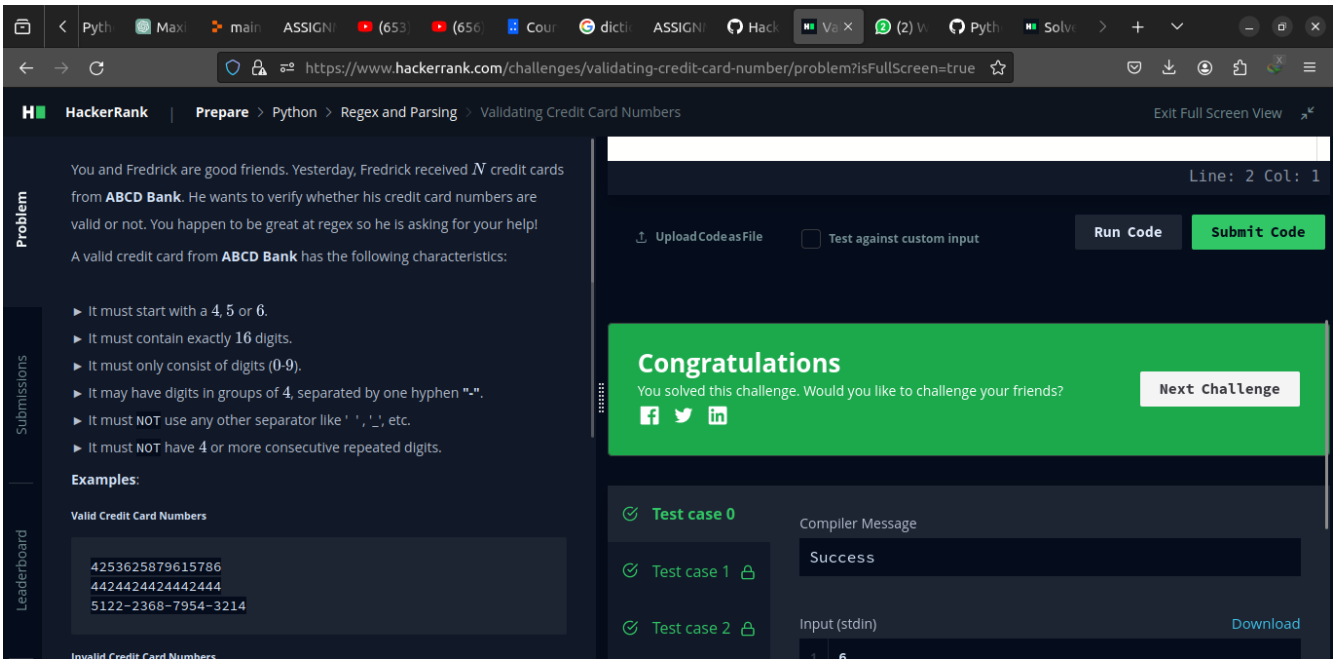
18- Regex Subsitution



Code

```
import re

def change(match):
symb = match.group(0)
if symb == "&&":
return "and"
elif symb == "||":
return "or"
n = int(input().strip())
for _ in range(n):
print(re.sub(r'(?<= )(&&|\|\|)(?= )', change, input()))
```

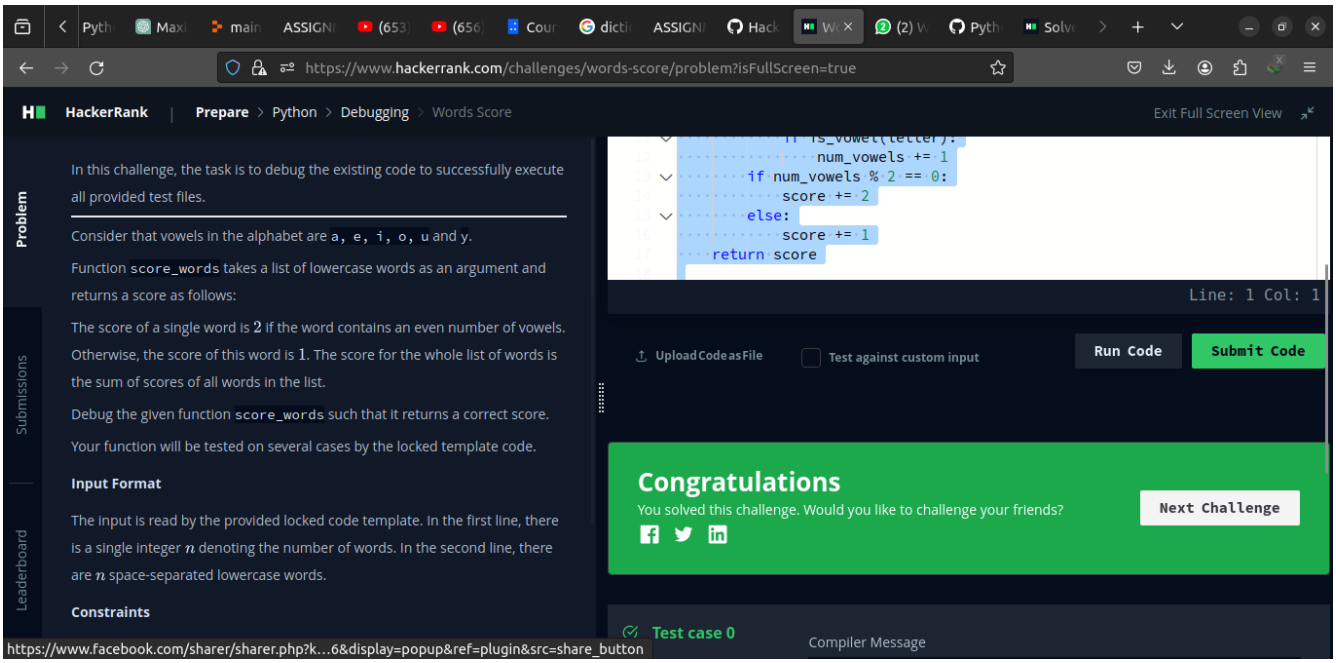
19- Validating Credit Card Numbers



Code

```
import re

if __name__ == "__main__":
t = int(input().strip())
for _ in range(t):
num = "".join(input())
if (re.match(r'^[456]', num) and
(re.match(r'([\d]{4}-){3}[\d]{4}$', num) or
re.match(r'[\d]{16}', num)) and
not re.search(r'(\d)\1{3,}', num.replace("-", ""))):
print("Valid")
else:
print("Invalid")
```

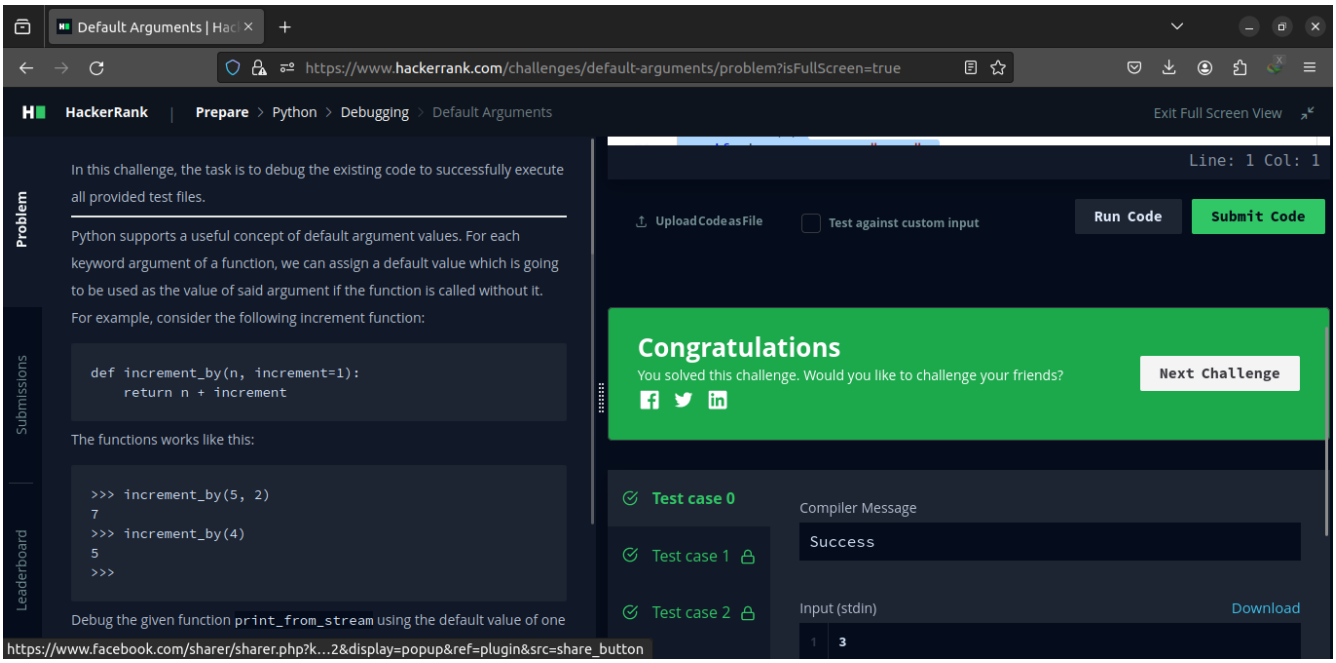
20- Word Score



Code

```
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))
```

21- Default Arguments

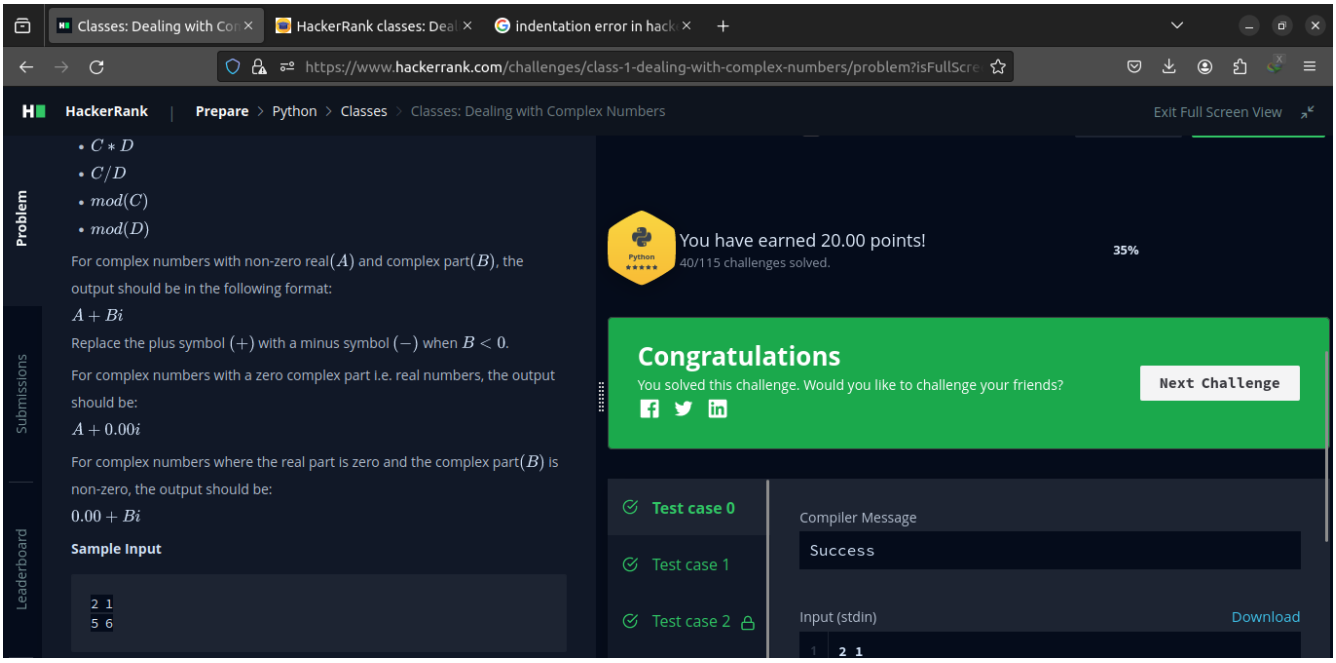


Code

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
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()):
temp = stream.current
for _ in range(n):
```

```
print(stream.get_next())
stream.current =temp


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())
```

22- Classes: Dealing with Complex Numbers

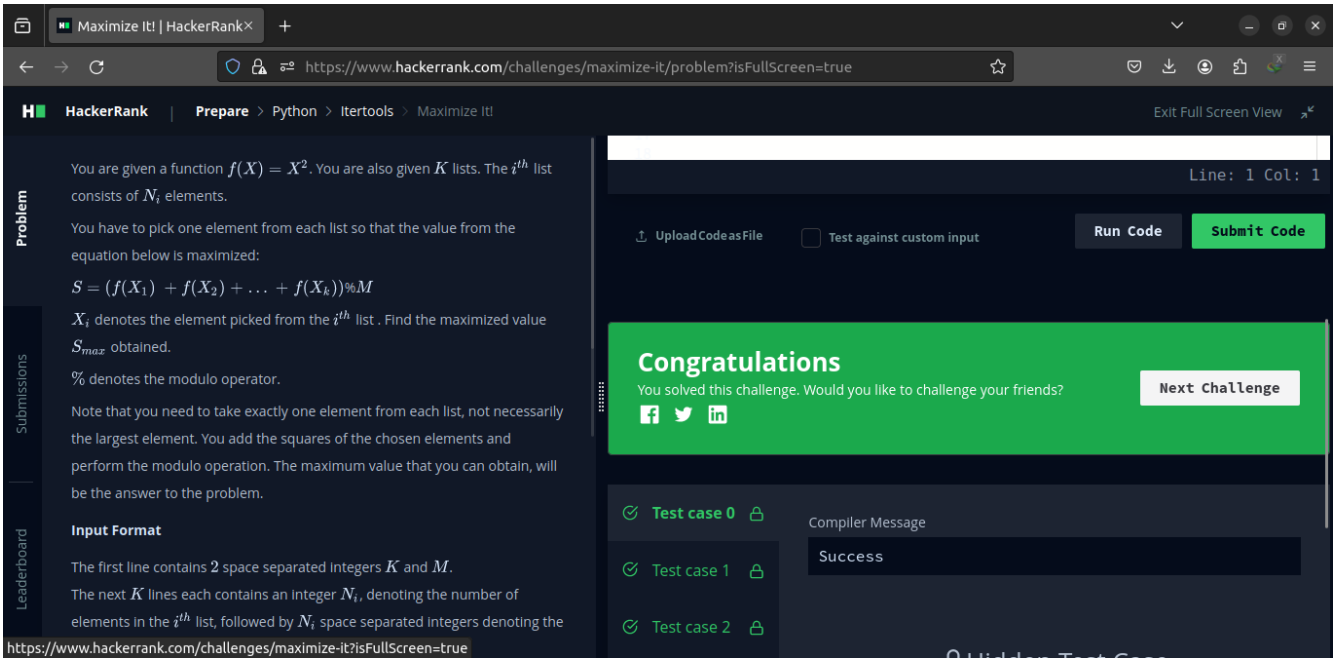


Code

```
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__':
```

“Hard Challenges”

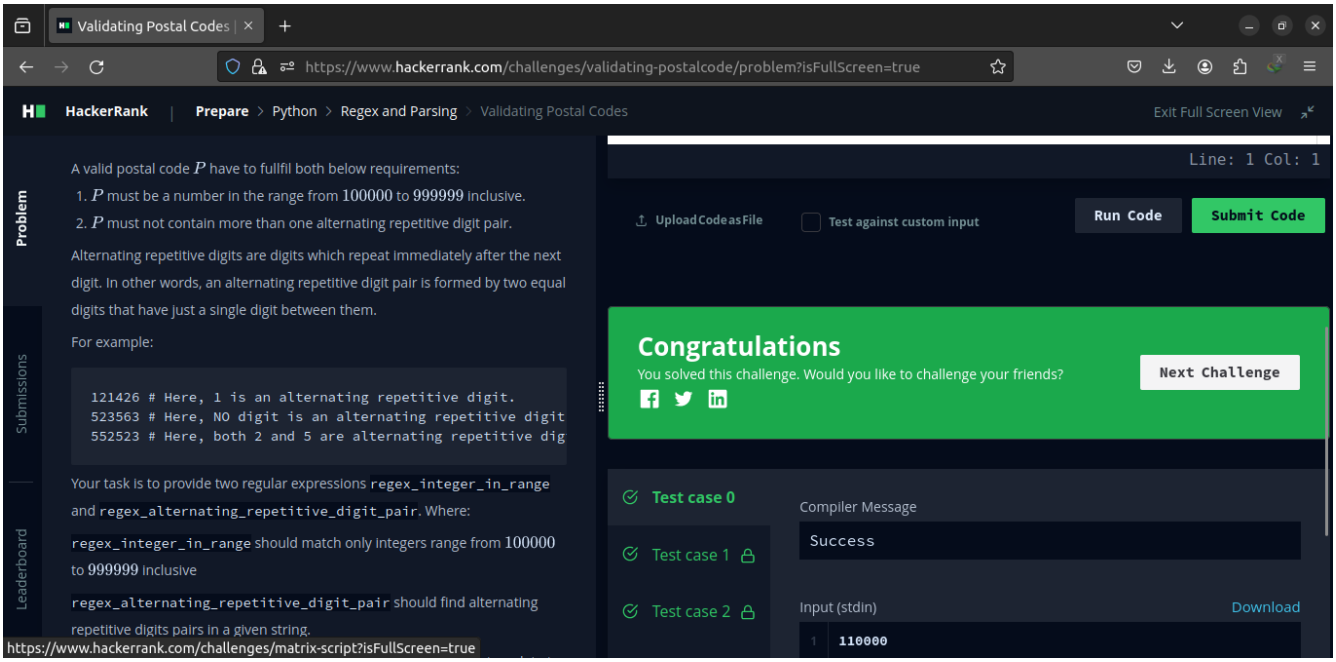
1- Maximize it



Code

```
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)
```

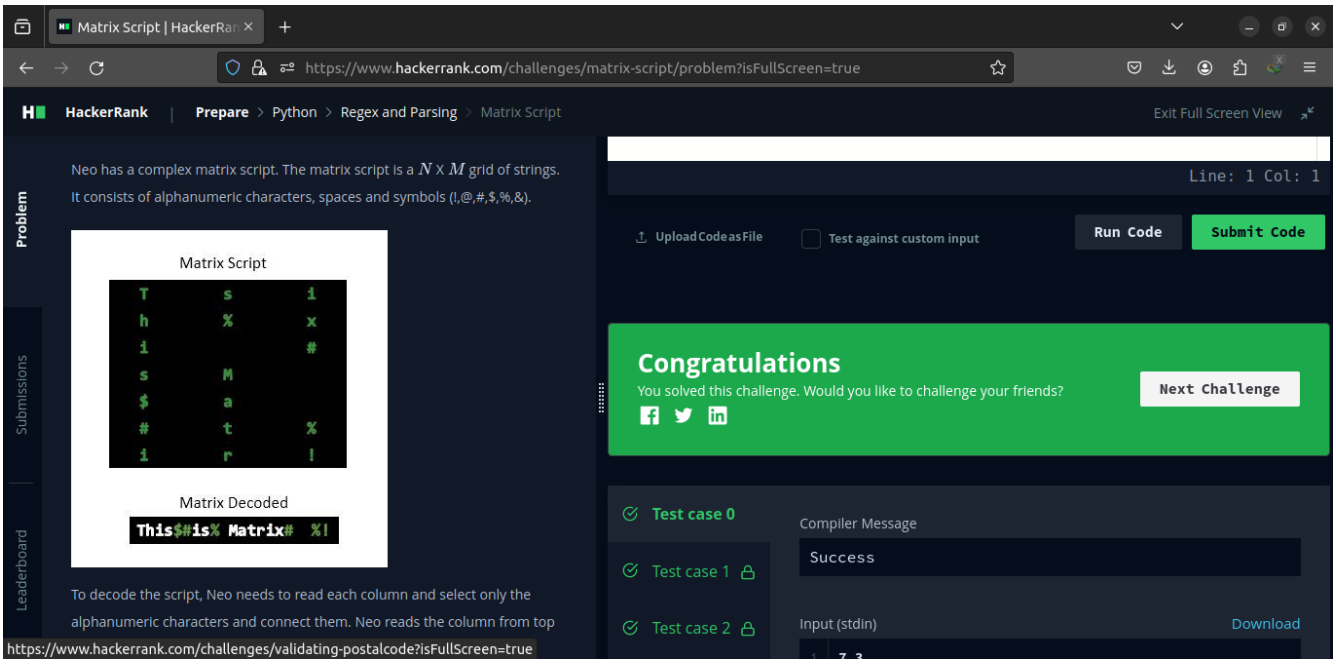
2- Validating Postal Address



Code

```
import re

num = input()
print(bool(re.match(r'^[1-9][\d]{5}$', num) and len(re.findall(r'(\d)(?=\d\1)', num))<2 ))
```

3- Matrix Script



Code

```
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))
```

Submitted to Sir Muhammad Affan