Assignment No 3

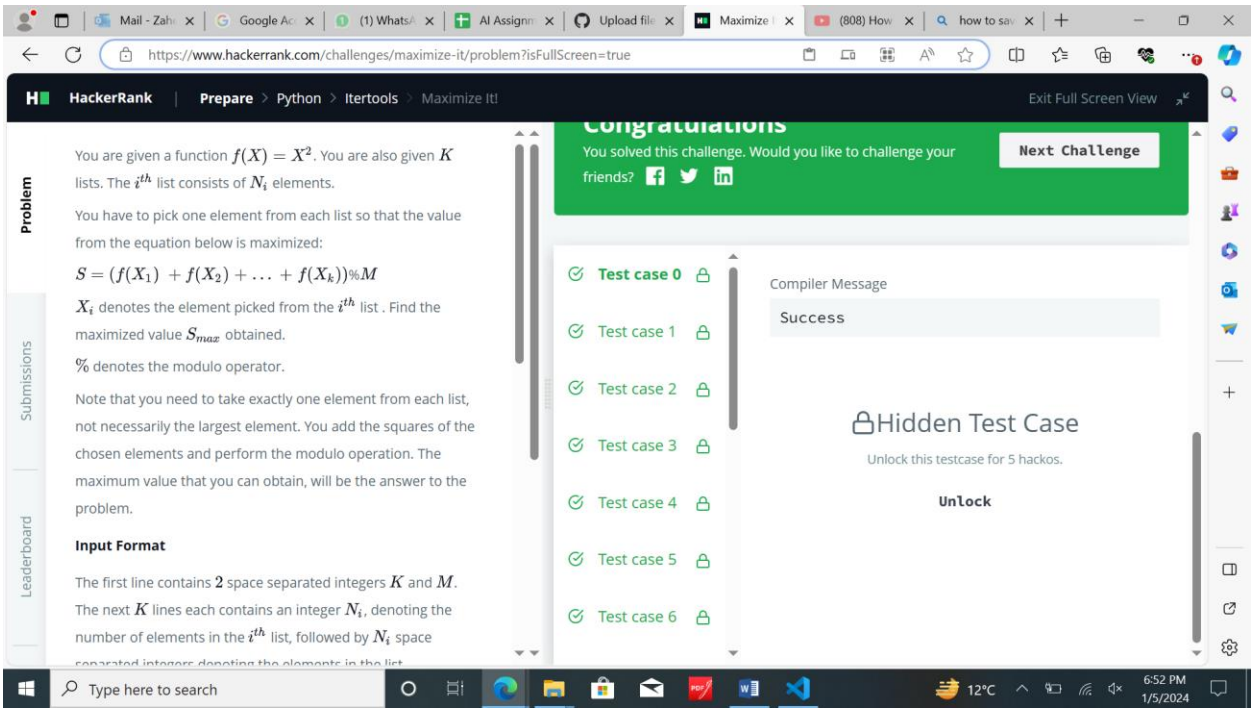
ZAHEER ABBAS

PHD MECHANICAL ENGINEERING

REG NO 433031

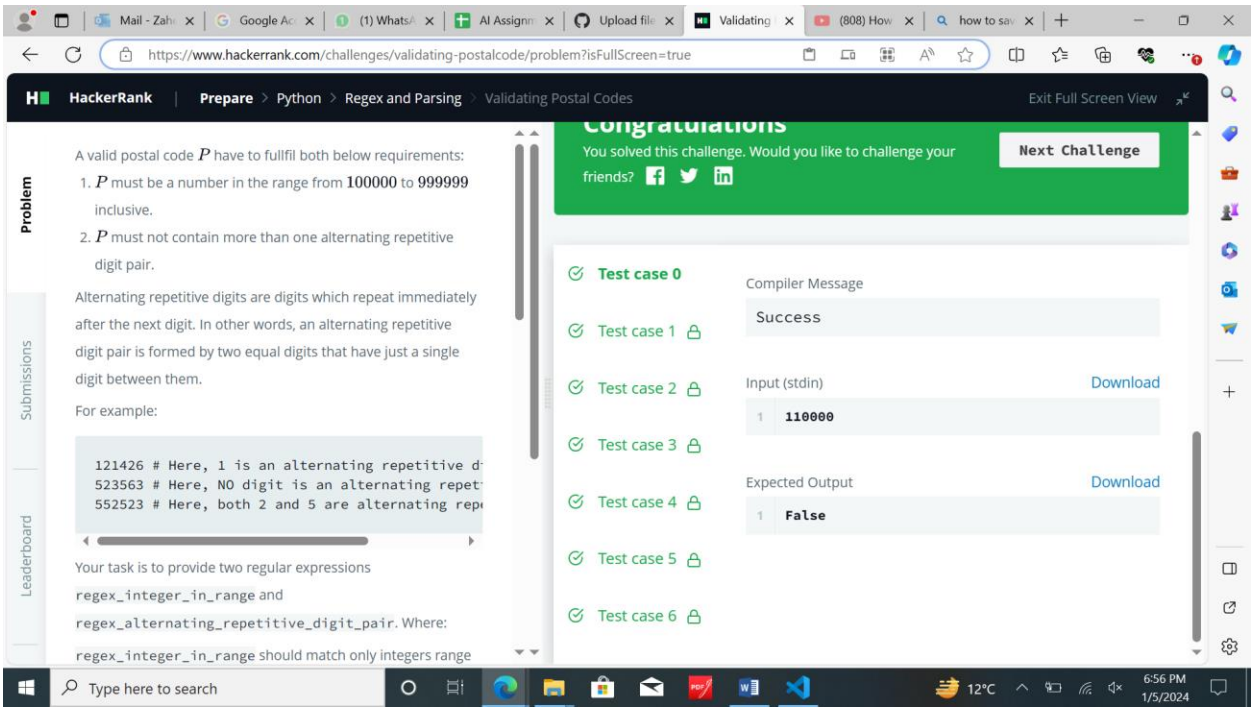
Maximize it – Hard

```
import itertools
line = input()
K = int(line.split()[0])
M = int(line.split()[1])
N = []
for i in range(K):
  l = input().split()
  l = [ int(n) for n in l ]
  l = l[1:]
  N.append(l)
pro = list( itertools.product( *N ) )
maxi = 0
for item in pro:
  sum=0
  for num in item:
    sum += num**2
  modu = sum % M
  if (modu > maxi):
    maxi = modu
print (maxi)
```



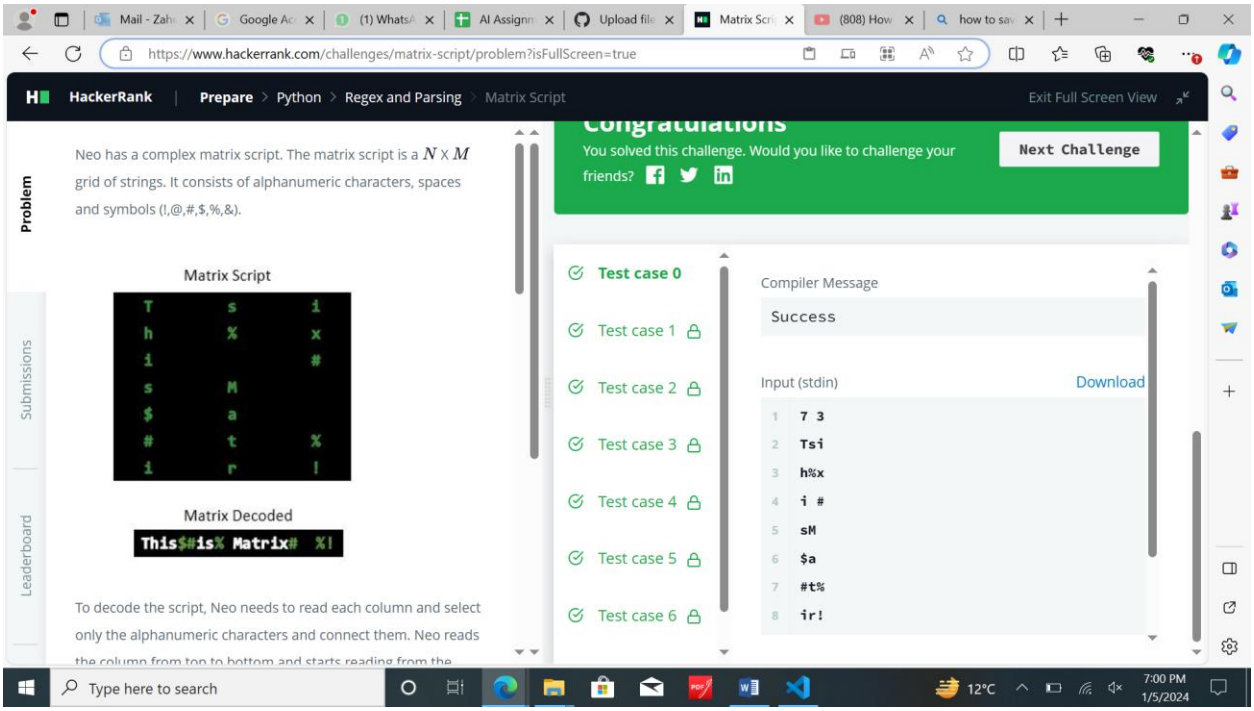
Validating Postal codes - Hard

```
regex_integer_in_range = r"^([1-9][0-9]{5})$"
regex_alternating_repetitive_digit_pair = r"(?=(\d)\d\1)"

import re
P = input()

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



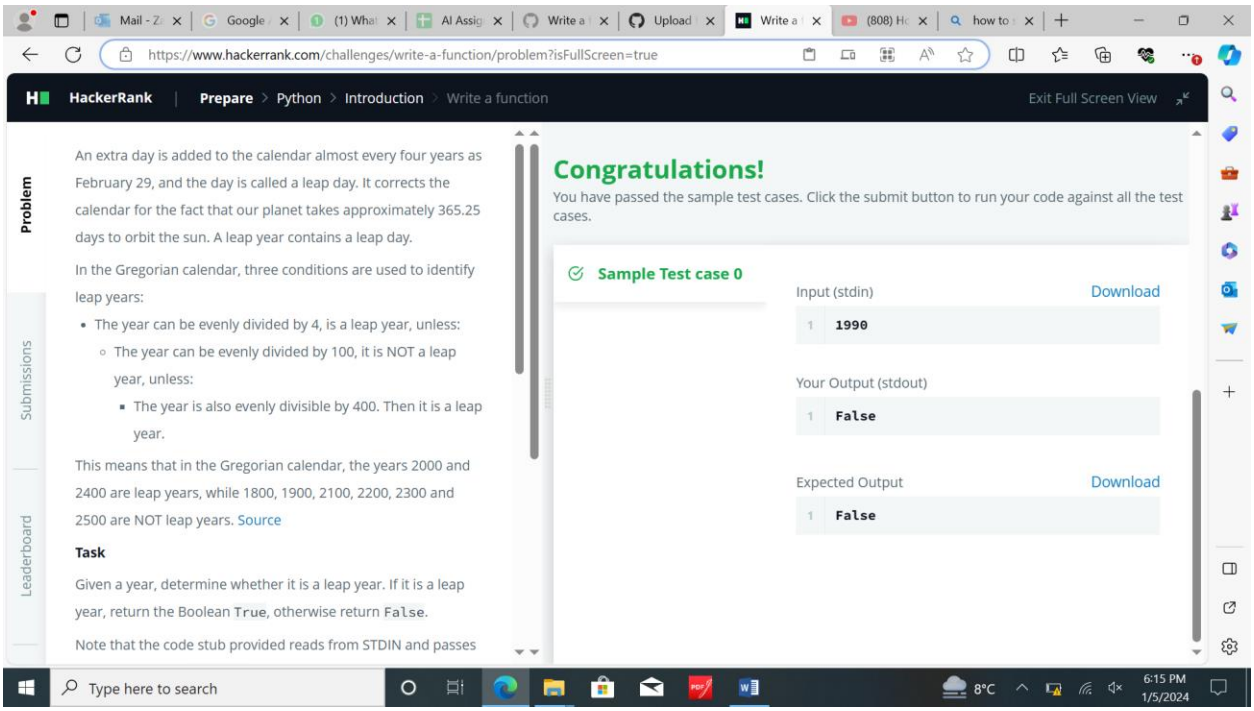
Matrix Script - Hard

```
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 = []
t = []
for _ in range(n):
    matrix_item = [x for x in input()]
    matrix.append(matrix_item)

for i in range(m):
    for j in range(n):
        t.append(matrix[j][i])
s = ''.join(t)
path = re.compile(r'\b[ !@#$%&]+\b', re.M)
k = re.sub(path, ' ', s)
print(k)
```



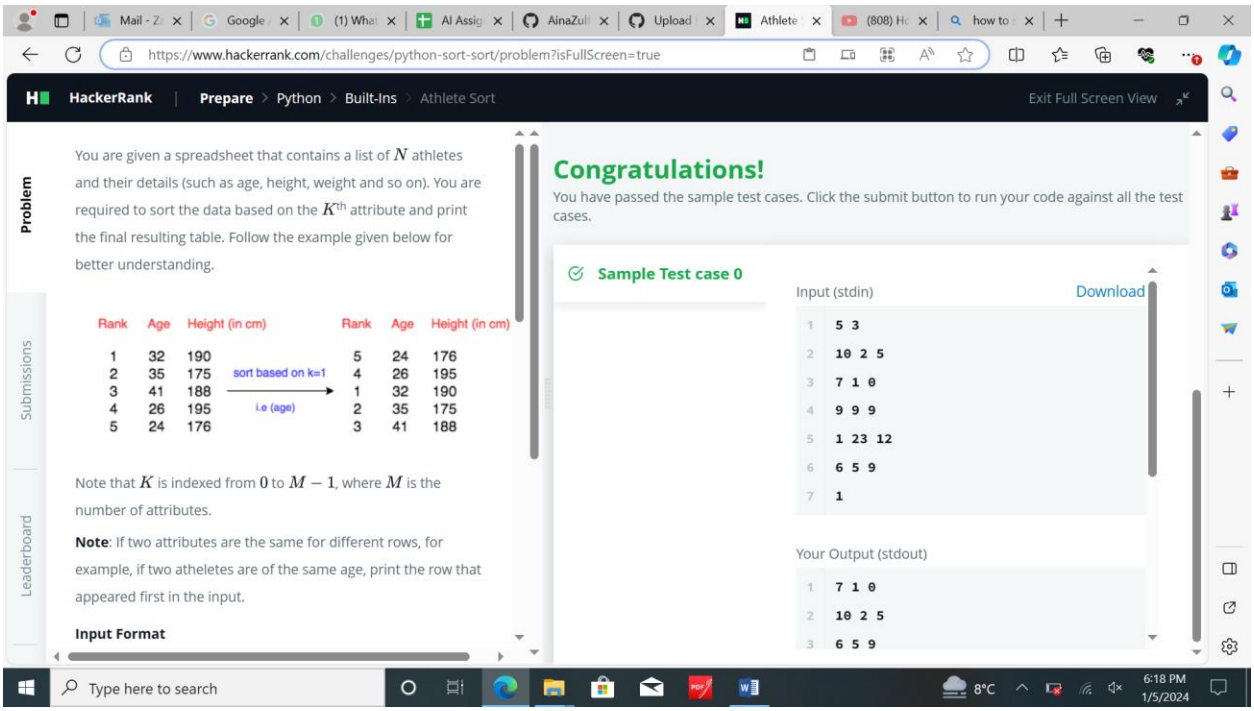
Write a Function (Leap Year)-Medium

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


    if 1900<= year <= 10**5:
        leap = False
        if year % 4 == 0 and year % 100 != 0:
            leap = True
        if year % 400 == 0:
            leap = True
    return leap



year = int(input())
```



Sort Athlete -Medium

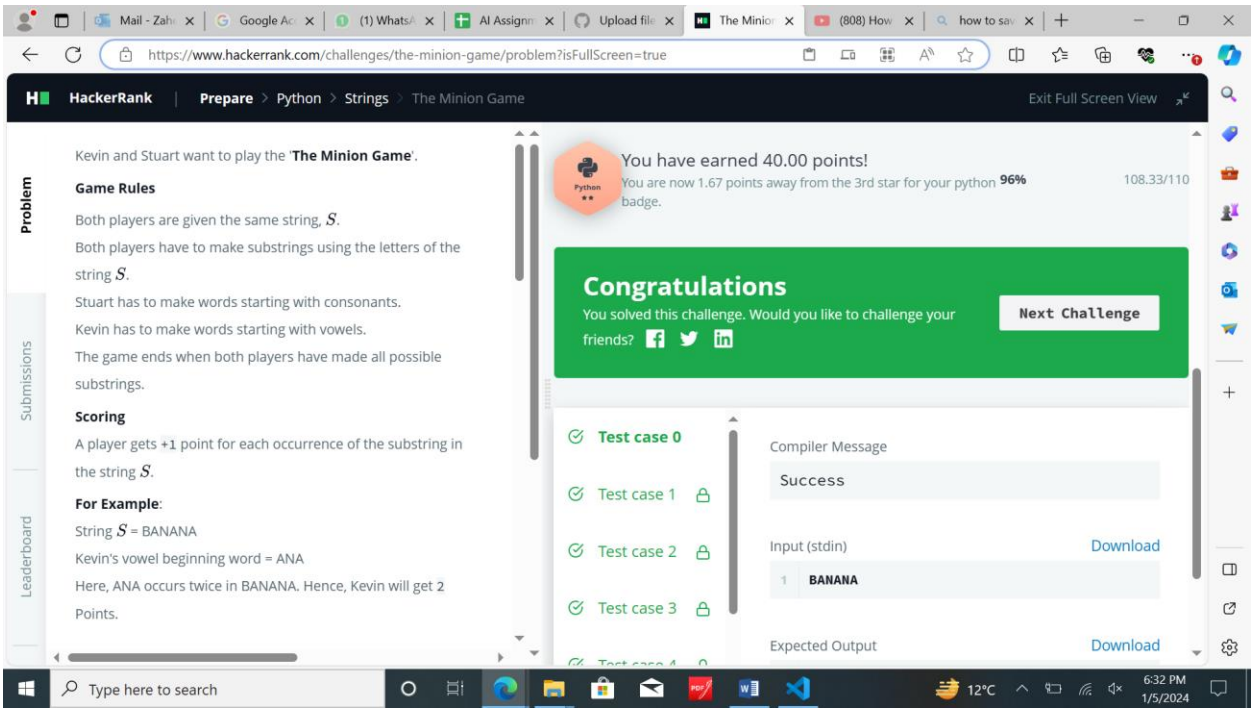
```
if __name__ == "__main__":
    n, m = input().strip().split(' ')
    n, m = [int(n), int(m)]
    arr = []
    for arr_i in range(n):
       arr_t = [int(arr_temp) for arr_temp in input().strip().split(' ')]
       arr.append(arr_t)
    k = int(input().strip())
    sorted_arr = sorted(arr, key = lambda x : x[k])
    for row in sorted_arr:
        print(' '.join(str(y) for y in row))
```



Minion Game -Medium

```python
def minion_game(string: str) -> None:
    kevin = stuart = 0
    length: int = len(string)
    for i, char in enumerate(string):
        points: int = length - i
        if char in {"A", "E", "I", "O", "U"}:
            kevin += points
        else:
            stuart += points
    if kevin == stuart:
        print("Draw")
```

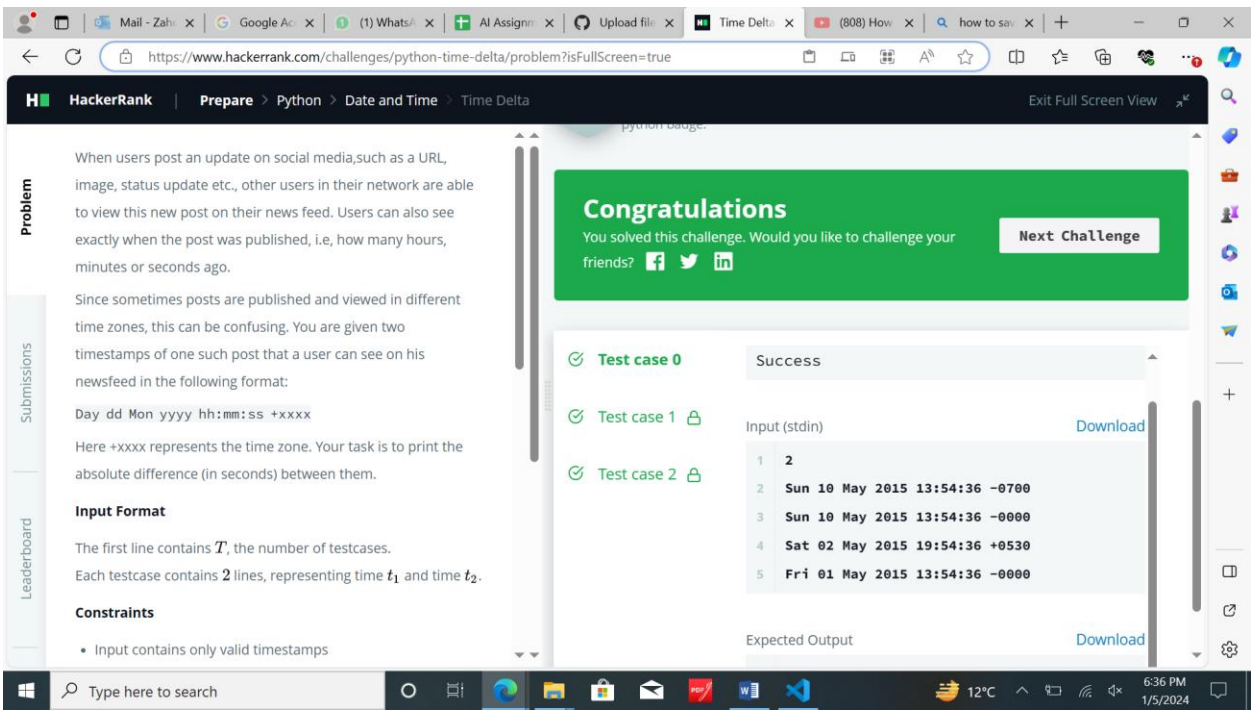
```
    else:
        print(*("Stuart", stuart) if stuart > kevin else ("Kevin", kevin))

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



Time Delta -Medium

```
import math
import os
import random
import re
import sys
from datetime import datetime
def time_delta(t1, t2):
    format_ = '%a %d %b %Y %H:%M:%S %z'
    t1 = datetime.strptime(t1, format_)
    t2 = datetime.strptime(t2, format_)
    return str(int(abs((t1-t2).total_seconds())))
if __name__ == '__main__':
```

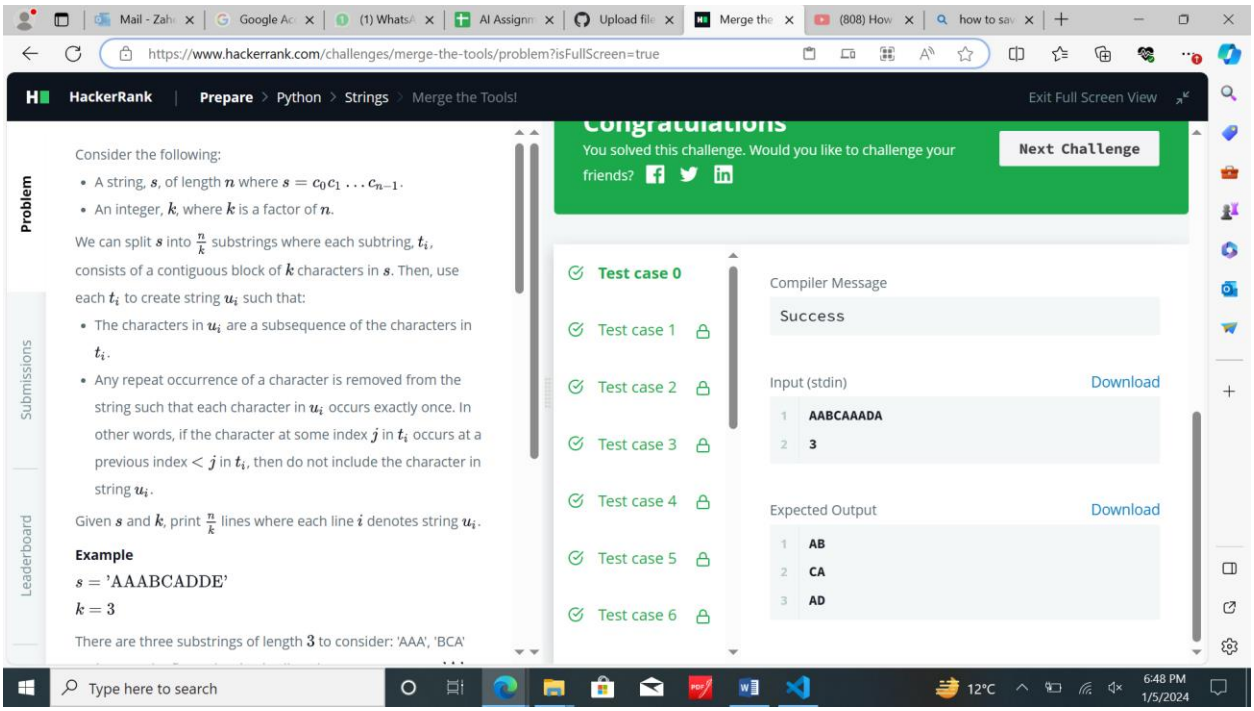
```python
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    t = int(input())

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

        t2 = input()

        delta = time_delta(t1, t2)

        fptr.write(delta + '\n')

    fptr.close()
```



Merge the tools! -Medium

```python
def merge_the_tools(string, k):
    l = len(string)//k
    for i in range(l):
        print(''.join(dict.fromkeys(string[i*k:(i*k)+k])))

if __name__ == '__main__':
```

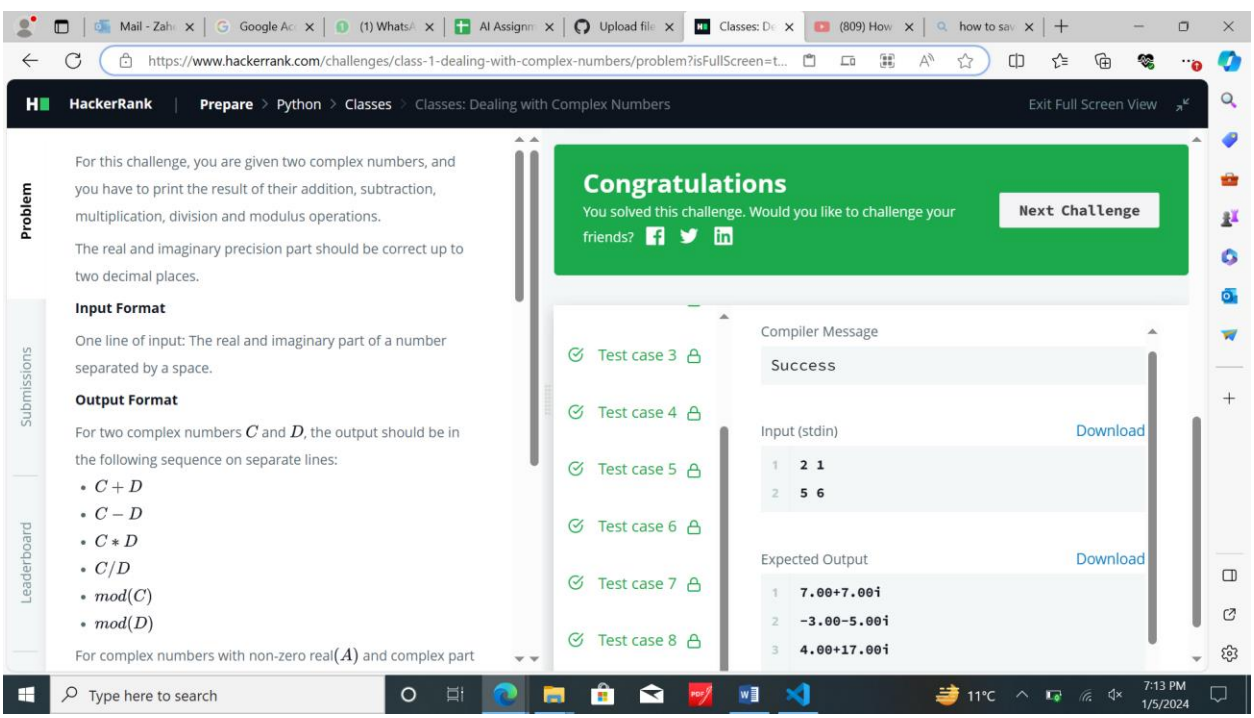
```
    string, k = input(), int(input())
    merge_the_tools(string, k)
```



Classes : dealing with complex numbers - Medium

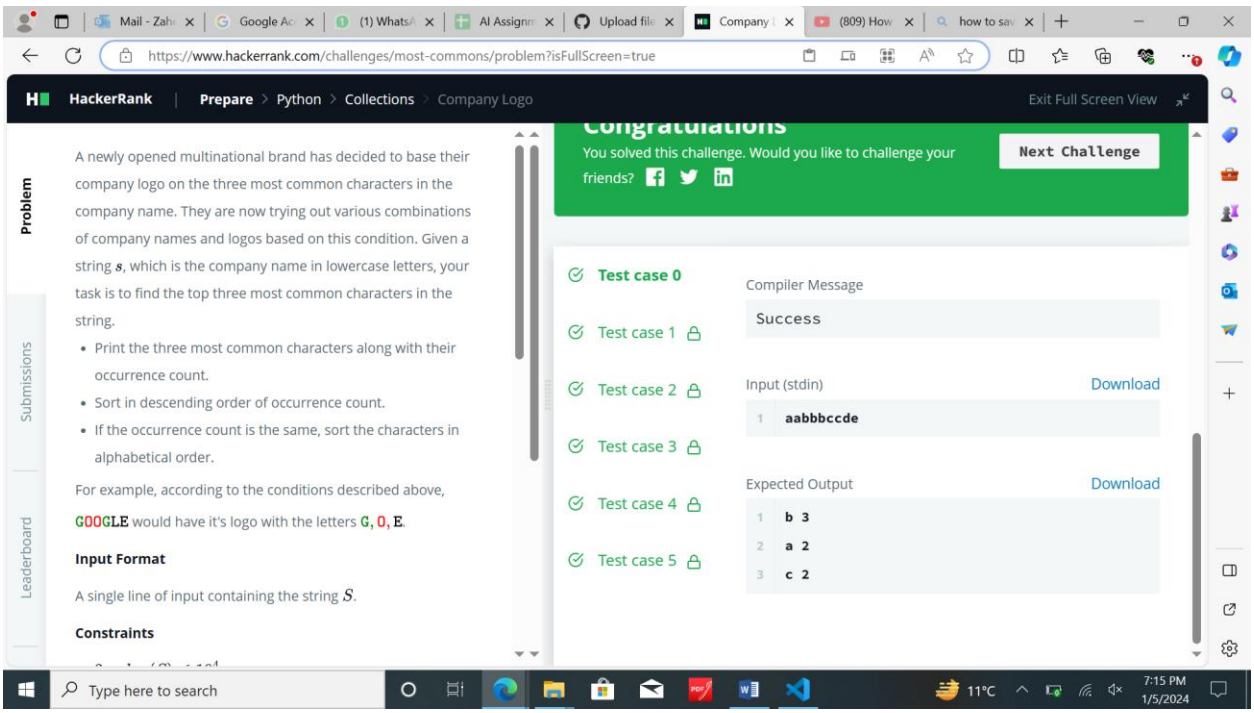
```
import math
class Complex(object):
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary
    def __add__(self, no):
        return Complex(self.real + no.real , self.imaginary + no.imaginary)
    def __sub__(self, no):
        return Complex(self.real - no.real , self.imaginary - no.imaginary)
    def __mul__(self, no):
        prod = complex(self.real , self.imaginary)*complex(no.real ,
no.imaginary)
        return Complex(prod.real , prod.imag)
    def __truediv__(self, no):
        div = complex(self.real , self.imaginary)/complex(no.real , no.imaginary)
        return Complex(div.real , div.imag)
    def mod(self):
        m = math.sqrt(self.real**2 + self.imaginary**2)
        return Complex(m,0)
```

```python
    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')
```



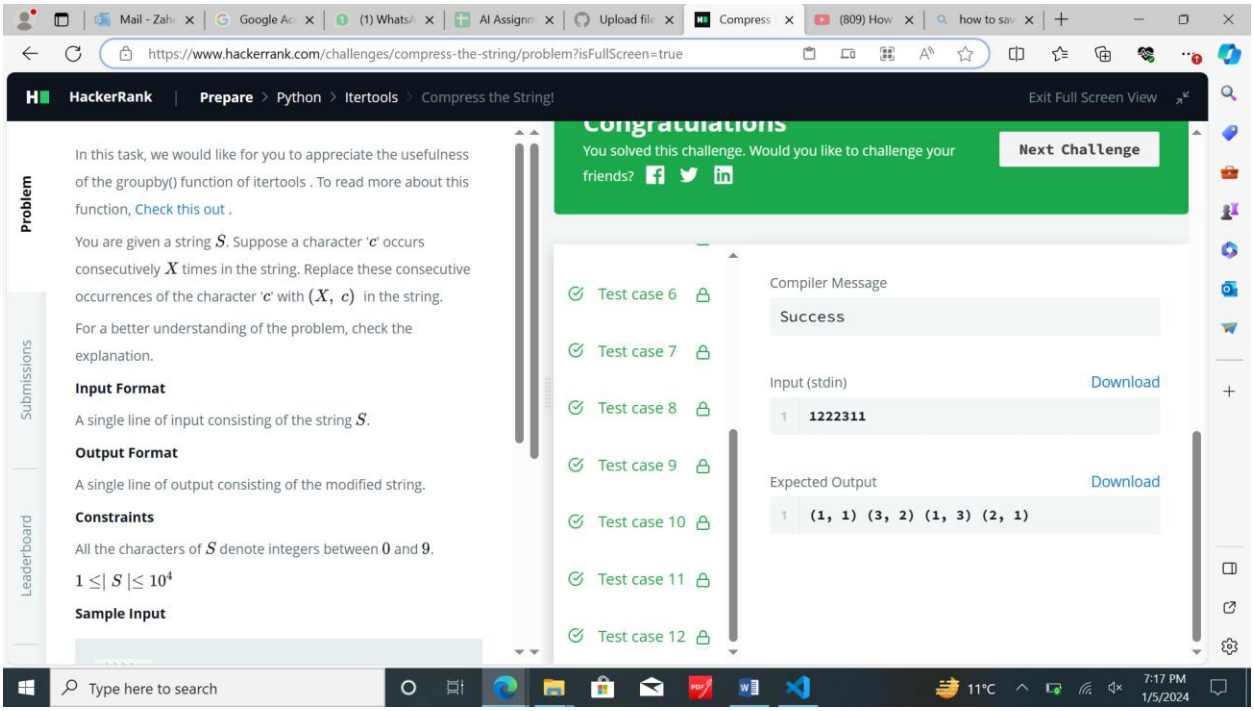
Company Logo - Medium

```python
import math
import os
import random
import re
import sys
from collections import Counter
if __name__ == '__main__':
    s = input()
    s = sorted(s)
    f = Counter(list(s))
    for k, v in f.most_common(3):
        print(k, v)
```



Compress the string - Medium

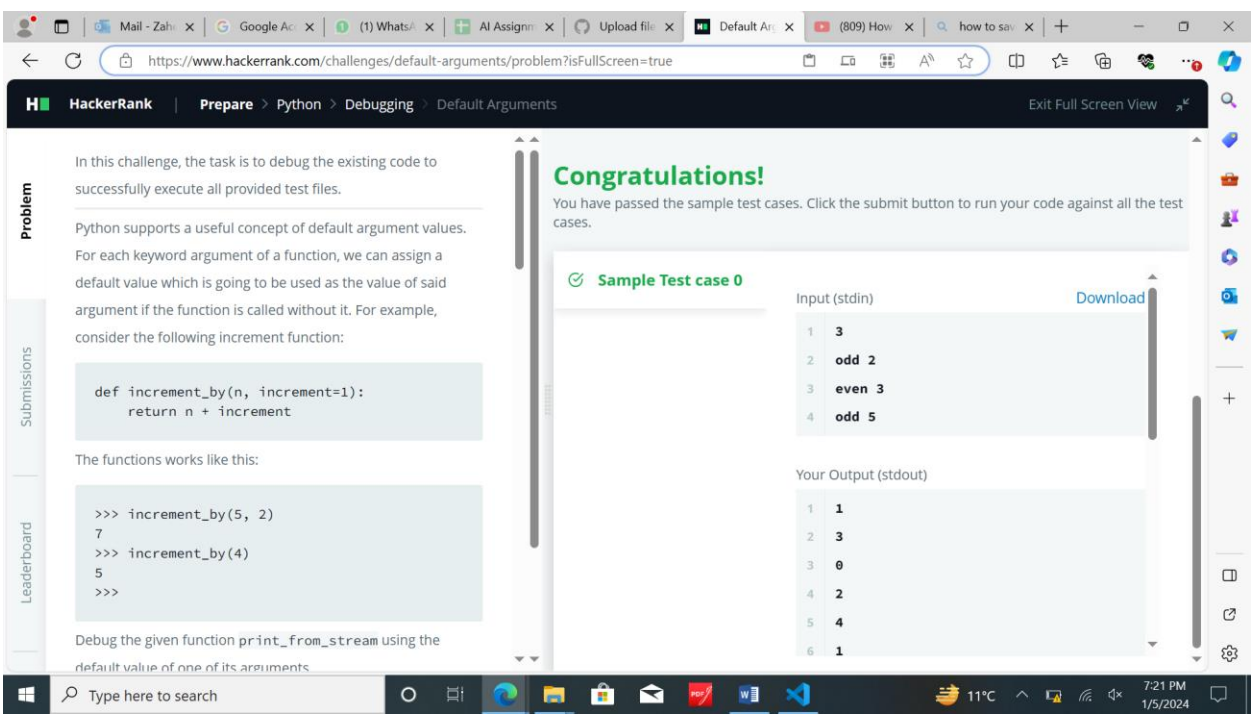
```
from itertools import groupby
for a, b in groupby(input()):
    print("(%d, %d)" % (len(list(b)), int(a)), end=' ')
```



Default Arguments - Medium

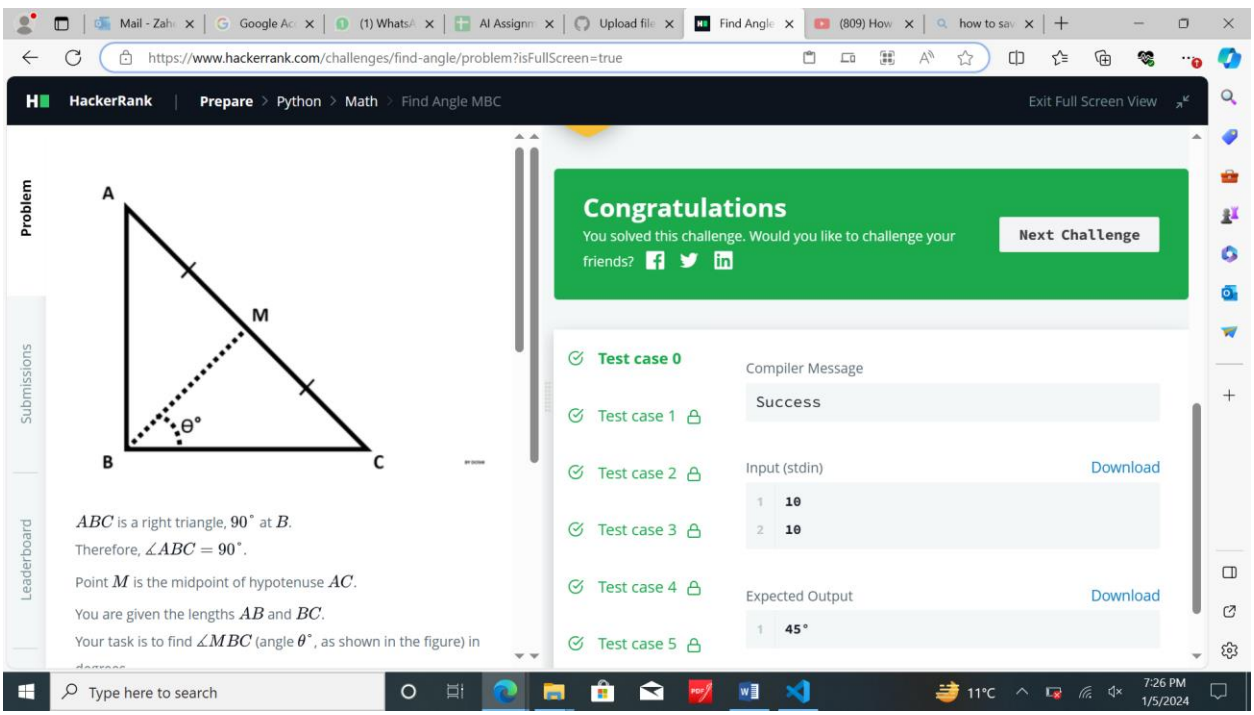
```
class EvenStream(object):
    def __init__(self):
        self.current = 0

    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

class OddStream(object):
    def __init__(self):
        self.current = 1
```

```python
    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

def print_from_stream(n, stream=None):
    if stream is None:
        stream = EvenStream()

    for _ in range(n):
        print(stream.get_next())
raw_input = input


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())
```



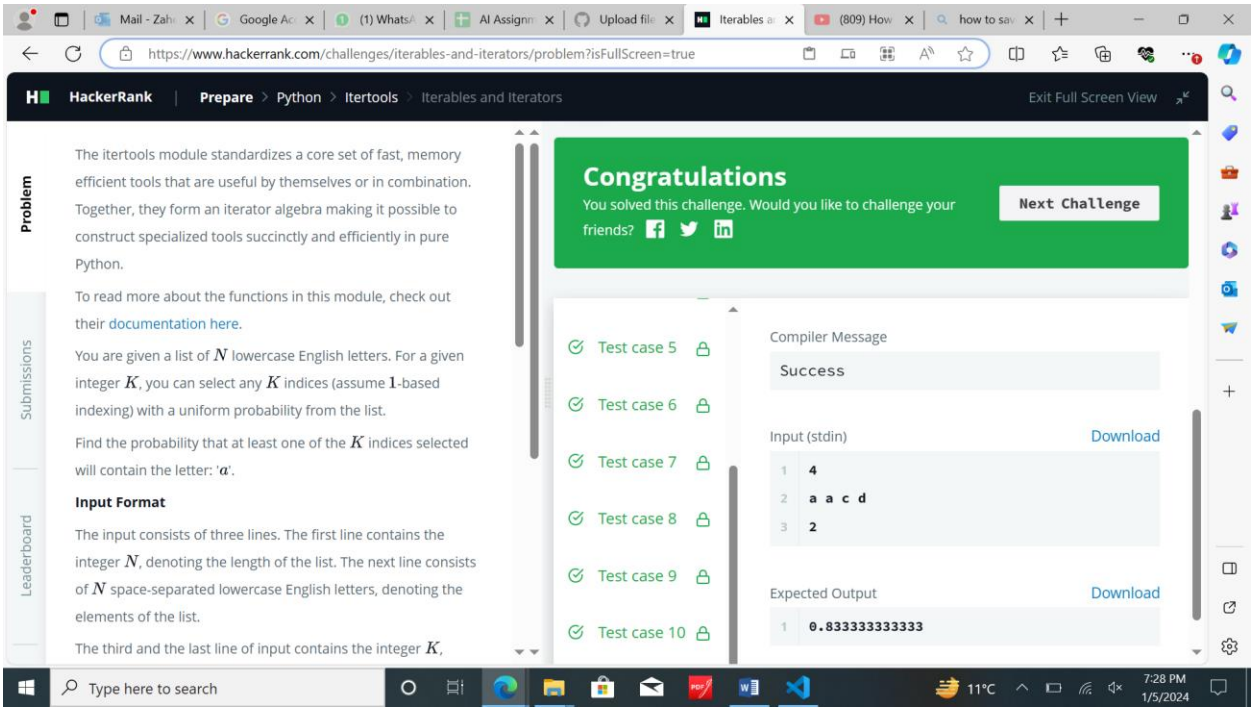
Finding Angle - Medium

```python
import math
def angle_MBC(AB, BC):
    theta_rad = math.atan2(AB, BC)
    theta_deg = round(theta_rad * (180 / math.pi))

    return theta_deg

AB = float(input())
BC = float(input())

angle = angle_MBC(AB, BC)
print("{}\u00b0".format(angle))
```



Iteratables and Iterators - Medium

```
from itertools import combinations, groupby

count, letters, select = int(input()), input().split(), int(input())
letters = sorted(letters)
combine = list(combinations(letters, select))
contain = len([c for c in combine if 'a' in c])
print(contain / len(combine))
```



No idea - Medium

```python
if __name__ == "__main__":
    happy = 0
    n, m = map(int, input().strip().split(' '))
    elements_arr = list(map(int, input().strip().split(' ')))

    A = set(map(int, input().strip().split(' ')))
    B = set(map(int, input().strip().split(' ')))

    for i in elements_arr:
        if i in A:
            happy += 1
        elif i in B:
            happy -= 1
    print(happy)
```