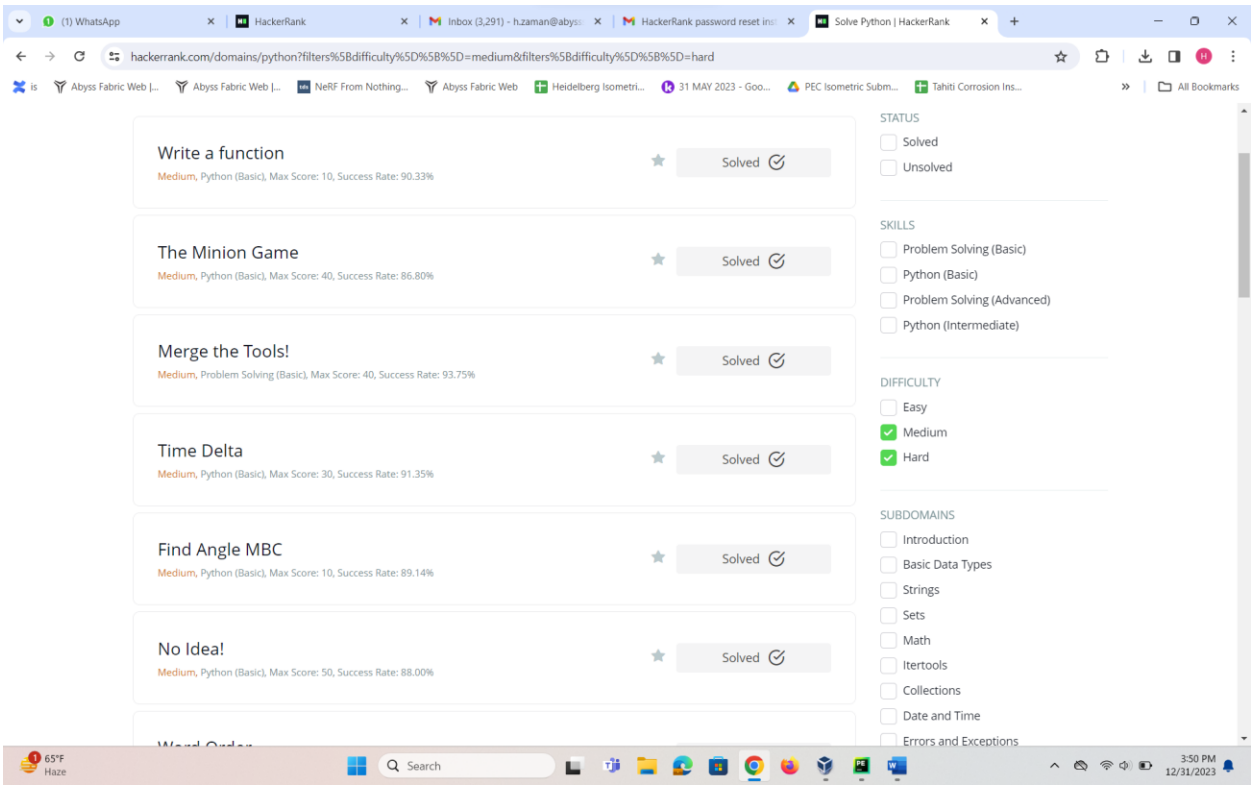
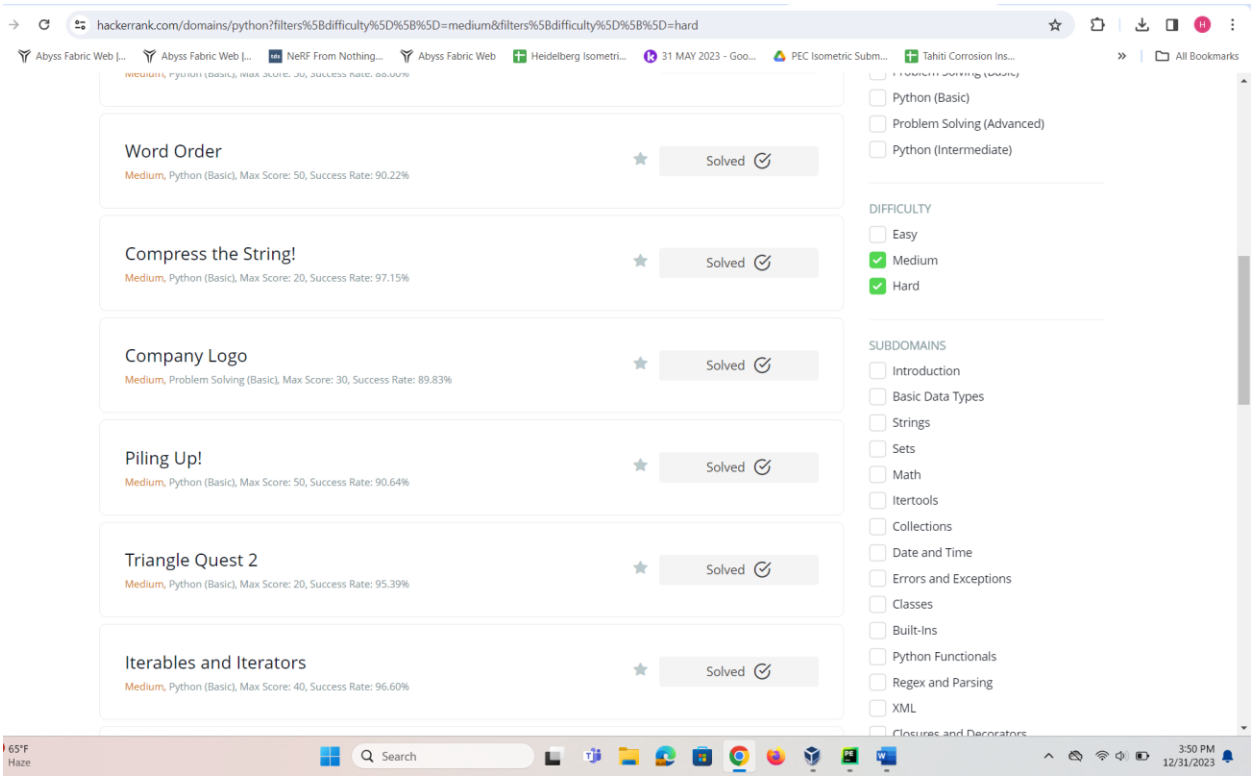
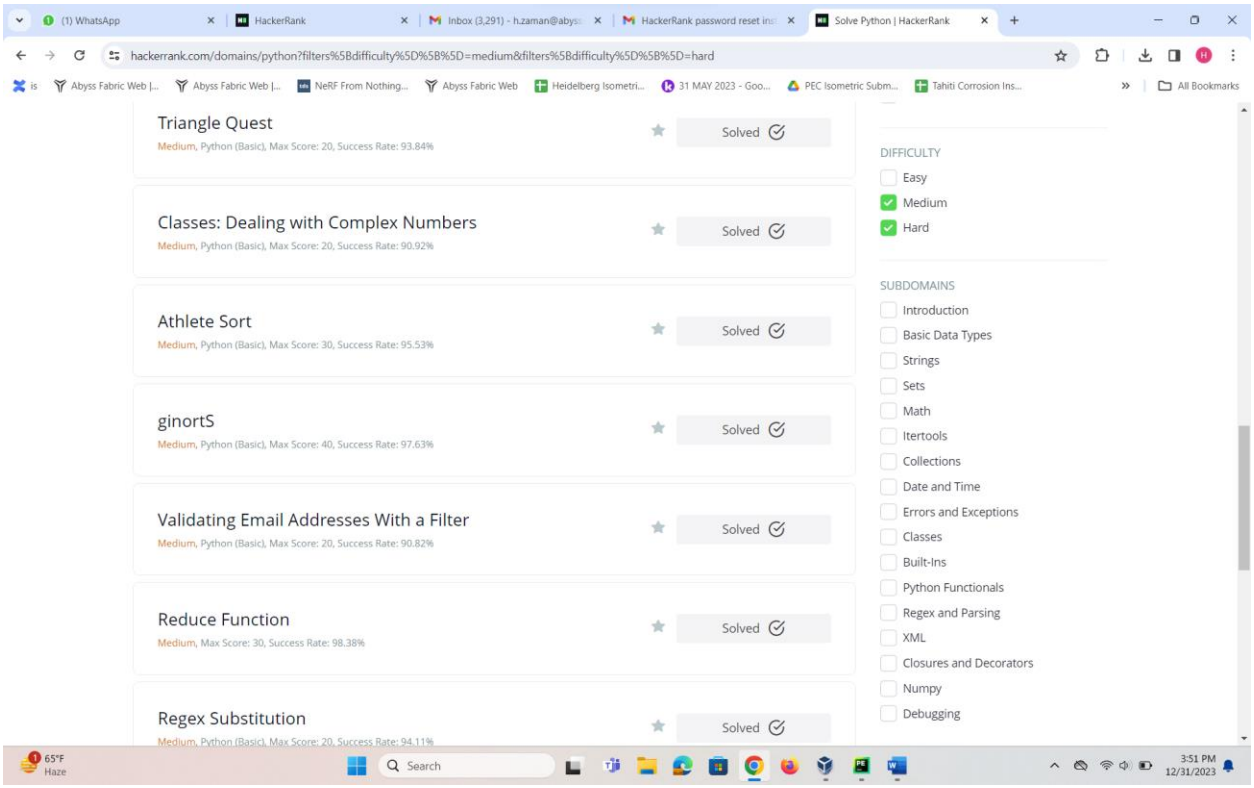
Name: Muhmmad Haris Zaman

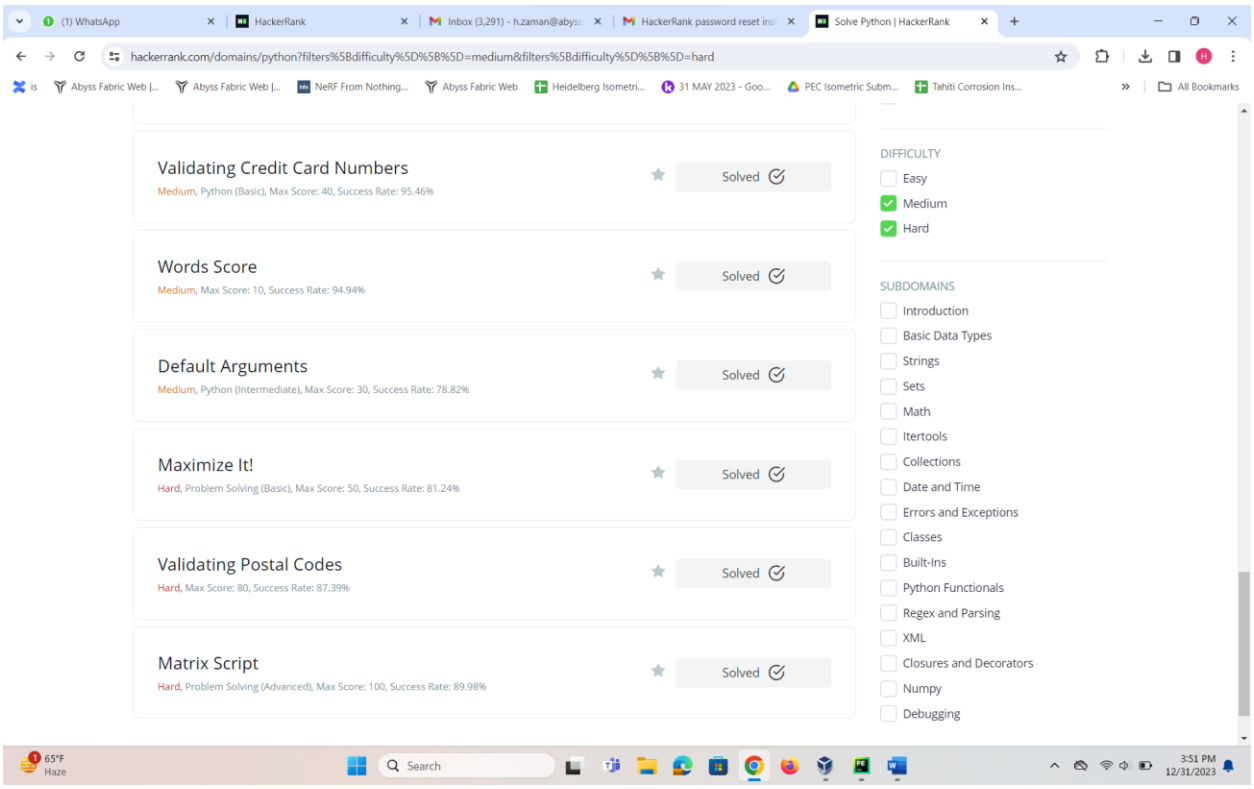
AI Assignment 3

Roll: 452133


Write a function
Medium, Python (Basic), Max Score: 10, Success Rate: 90.33%
Solved
The Minion Game
Medium, Python (Basic), Max Score: 40, Success Rate: 86.80%
Solved
Merge the Tools!
Medium, Problem Solving (Basic), Max Score: 40, Success Rate: 93.75%
Solved
Time Delta
Medium, Python (Basic), Max Score: 30, Success Rate: 91.35%
Solved
Find Angle MBC
Medium, Python (Basic), Max Score: 10, Success Rate: 89.14%
Solved
No Idea!
Medium, Python (Basic), Max Score: 50, Success Rate: 88.00%
Solved
STATUS
Solved
Unsolved
SKILLS
Problem Solving (Basic)
Python (Basic)
Problem Solving (Advanced)
Python (Intermediate)
DIFFICULTY
Easy
Medium
Hard
SUBDOMAINS
Introduction
Basic Data Types
Strings
Sets
Math
Itertools
Collections
Date and Time
Errors and Exceptions


Word Order
Medium, Python (Basic), Max Score: 50, Success Rate: 90.22%
Solved
Compress the String!
Medium, Python (Basic), Max Score: 20, Success Rate: 97.15%
Solved
Company Logo
Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.83%
Solved
Piling Up!
Medium, Python (Basic), Max Score: 50, Success Rate: 90.64%
Solved
Triangle Quest 2
Medium, Python (Basic), Max Score: 20, Success Rate: 95.39%
Solved
Iterables and Iterators
Medium, Python (Basic), Max Score: 40, Success Rate: 96.60%
Solved
Python (Basic)
Problem Solving (Advanced)
Python (Intermediate)
DIFFICULTY
Easy
Medium
Hard
SUBDOMAINS
Introduction
Basic Data Types
Strings
Sets
Math
Itertools
Collections
Date and Time
Errors and Exceptions
Classes
Built-Ins
Python Functionals
Regex and Parsing
XML


(1) WhatsApp
HackerRank
Inbox (3,291) - h.zaman@abyss
HackerRank password reset ins
Solve Python | HackerRank
hackerrank.com/domains/python?filters%5Bdifficulty%5D%5B%5D=medium&filters%5Bdifficulty%5D%5B%5D=hard
Abyss Fabric Web |...
Abyss Fabric Web |...
NeRF From Nothing...
Abyss Fabric Web
Heidelberg Isometri...
31 MAY 2023 - Goo...
PEC Isometric Subm...
Tahiti Corrosion Ins...
All Bookmarks
Triangle Quest
Medium, Python (Basic), Max Score: 20, Success Rate: 93.84%
Solved
Classes: Dealing with Complex Numbers
Medium, Python (Basic), Max Score: 20, Success Rate: 90.92%
Solved
Athlete Sort
Medium, Python (Basic), Max Score: 30, Success Rate: 95.53%
Solved
ginortS
Medium, Python (Basic), Max Score: 40, Success Rate: 97.63%
Solved
Validating Email Addresses With a Filter
Medium, Python (Basic), Max Score: 20, Success Rate: 90.82%
Solved
Reduce Function
Medium, Max Score: 30, Success Rate: 98.38%
Solved
Regex Substitution
Medium, Python (Basic), Max Score: 20, Success Rate: 94.11%
Solved
DIFFICULTY
Easy
Medium
Hard
SUBDOMAINS
Introduction
Basic Data Types
Strings
Sets
Math
Itertools
Collections
Date and Time
Errors and Exceptions
Classes
Built-Ins
Python Functionals
Regex and Parsing
XML
Closures and Decorators
Numpy
Debugging
65°F
Haze
Search
3:51 PM
12/31/2023


(1) WhatsApp
HackerRank
Inbox (3,291) - h.zaman@abyss
HackerRank password reset ins
Solve Python | HackerRank
hackerrank.com/domains/python?filters%5Bdifficulty%5D%5B%5D=medium&filters%5Bdifficulty%5D%5B%5D=hard
Abyss Fabric Web |...
Abyss Fabric Web |...
NeRF From Nothing...
Abyss Fabric Web
Heidelberg Isometri...
31 MAY 2023 - Goo...
PEC Isometric Subm...
Tahiti Corrosion Ins...
All Bookmarks
Validating Credit Card Numbers
Medium, Python (Basic), Max Score: 40, Success Rate: 95.46%
Solved
Words Score
Medium, Max Score: 10, Success Rate: 94.94%
Solved
Default Arguments
Medium, Python (Intermediate), Max Score: 30, Success Rate: 78.82%
Solved
Maximize It!
Hard, Problem Solving (Basic), Max Score: 50, Success Rate: 81.24%
Solved
Validating Postal Codes
Hard, Max Score: 80, Success Rate: 87.39%
Solved
Matrix Script
Hard, Problem Solving (Advanced), Max Score: 100, Success Rate: 89.98%
Solved
DIFFICULTY
Easy
Medium
Hard
SUBDOMAINS
Introduction
Basic Data Types
Strings
Sets
Math
Itertools
Collections
Date and Time
Errors and Exceptions
Classes
Built-Ins
Python Functionals
Regex and Parsing
XML
Closures and Decorators
Numpy
Debugging
65°F
Haze
Search
3:51 PM
12/31/2023

Code 1: Minion Game

```
def minion_game(string):
    # your code goes here
    length = len(string)
    players=["Stuart","Kevin"]
    consonentCount=0
    vowelCount=0
    for i in range(length):
        if string[i] in "AEIOU":
            vowelCount+= length-i
        else:
            consonentCount+= length-i
    if consonentCount>vowelCount:
        print(f"{players[0]} {consonentCount}")
    elif consonentCount<vowelCount:
        print(f"{players[1]} {vowelCount}")
    else:
        print("Draw")
```

Code 2: Merge the tool

```
def merge_the_tools(string, k):
    # your code goes here

    temporary=""
    for i in range(len(string)):
        if string[i] not in temporary:
            temporary+= string[i]

        if (i+1)%k==0:
            print(temporary)
            temporary=""
```

Code 3: gnortS

```
# Enter your code here. Read input from STDIN. Print output to STDO
UT
string = input()
uCase,lCase,oddDigits,evenDigits="","","",""
for x in string:
    if x.isalpha():
        if x.islower():
```

```
            lCase+=x
        else:
            uCase+=x
    elif x.isdigit():
            if int(x)%2==0:
                evenDigits+=x
            else:
                oddDigits+=x

result=sorted(lCase)+sorted(uCase)+sorted(oddDigits)+sorted(evenDig
its)
print("".join(result))
```

Code 4: Time delta

```
#!/bin/python3


import math

import os

import random

import re

import sys

import time

from datetime import datetime, timedelta


def time_delta(t1, t2):

    format_string = "%a %d %b %Y %H:%M:%S %z"

    dt1 = datetime.strptime(t1, format_string)

    dt2 = datetime.strptime(t2, format_string)

    diff = abs(dt1 - dt2)

    return str(int(diff.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()
```

Code 5: 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)
s=""
```

```
for i in range(m):
    for j in range(n):
        s+=matrix[j][i]
pattern=r"(?<=[a-zA-Z0-9])[\s!@#$%&]+(?=[a-zA-Z0-9])"
s=re.sub(pattern, " ", s)
print(s)
```

Code 6: find angle mbc

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

import math

AB = int(input().strip())
BC = int(input().strip())
radians = math.atan(AB/BC)
deg = round(math.degrees(radians))
print(f'{deg}' + u'\N{DEGREE SIGN}')

Code 18 no idea
input()
arr = list(map(int, input().split()))
a = set(map(int, input().split()))
b = set(map(int, input().split()))
happiness = 0

for i in arr:
    if i in a:
        happiness+=1
```

```
    if i in b:
        happiness-=1

print(happiness)
```

Code 7: Word order

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import defaultdict
d = defaultdict(int)
for _ in range(int(input())):
    d[input()] += 1
print(len(d))
for i in d: print(d[i],end=' ')
```

Code 8: compress the string

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import groupby

s, funlis = input(), []

for key, group in groupby(s):
    count = len(list(group))
    funlis.append((count, int(key)))

for item in funlis:
    print(item, end = ' ')
```

Code 9: company logo

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



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


s_list=[i for  i in s]
s_set=list(set(s_list))
s_dict={}
for i in s_set:
    s_dict[i]=s_list.count(i)


sorted_dict={key: val for key, val in sorted(s_dict.items(), key = lambda ele
: (-ele[1],ele[0]))}
c=0
for k,v in sorted_dict.items():
    print(k+" "+str(v))
    c=c+1
    if (c>=3):
        break
```

Code 10: piling up

```
for _ in range(int(input())):
    l = int(input())
    a = list(map(int, input().split()))
```

```
    left = 0
    right = l - 1
    last = float('inf')
    curr = True

    while left <= right:
        if a[left] >= a[right]:
            if a[left] <= last:
                last = a[left]
                left += 1
            else:
                curr = False
                break
        else:
            if a[right] <= last:
                last = a[right]
                right -= 1
            else:
                curr = False
                break

    print("Yes" if curr else "No")
```

Code 11: triangle quest 2

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

Code 12: iterable and iterators

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations

N, letters, k = int(input()), input().split(), int(input())
com = list(combinations(letters, k))
count = 0

for c in com:
    if 'a' in c:
        count += 1

print("%.3f" % (count/len(com)))
```

Code 13: triangle quest

```
for i in range(1, int(input())):
     print(ascii(i)*i)
```

Code 14: classes dealing with complexity

```
import math

import math

class Complex(object):
    def __init__(self, real, imaginary):
        self.number = complex(real, imaginary)
        self.real = self.number.real
```

```
        self.imaginary = self.number.imag

    def __add__(self, no):
        res = self.number + complex(no.real, no.imaginary)
        return Complex(res.real, res.imag)

    def __sub__(self, no):
        res = self.number - complex(no.real, no.imaginary)
        return Complex(res.real, res.imag)

    def __mul__(self, no):
        res = self.number * complex(no.real, no.imaginary)
        return Complex(res.real, res.imag)

    def __truediv__(self, no):
        res = self.number / complex(no.real, no.imaginary)
        return Complex(res.real, res.imag)

    def mod(self):
        res = abs(self.number)
        return Complex(res.real, res.imag)

    def __str__(self):
        if self.imaginary == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            result = "0.00+%.2fi" % (self.imaginary) if self.imaginary >= 0 e
lse "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')
```

Code 15: Maximize it

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import product

K, M = map(int, input().split())
k_lines=[]

for _ in range(K):
    k_line = list(map(int, input().split()))
    k_lines.append(k_line[1:])

s_max=0
```

```
for s_ in product(*k_lines):
  current_s_max = sum(x**2 for x in s_) % M
  if current_s_max > s_max:
        s_max = current_s_max

print(s_max)
```

Code 16: validating postal codes

```
regex_integer_in_range = r"^[1-9]\d{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)
```

Code 17: validating email

```
import re
def fun(s):
    pattern = re.compile(r'^[a-zA-Z0-9_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$')
    return pattern.match(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)
```

Code 18: Reduce Function

```
from fractions import Fraction
from functools import reduce
def product(fracs):
    t = Fraction(reduce(lambda x,y: x * y, fracs))
    return (t.numerator, t.denominator)
```

Code 19: Regex substitution

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
n = int(input())
string = '\n'.join([input() for _ in range(n)])
print(re.sub(r"(?<=(\s))\|\|(?=(\s))", 'or', re.sub(r"(?<=(\s))&&(?=(\s))", '
and', string)))
```

Code 20: validating Credit card Numbers

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
N = int(input())
for _ in range(N):
```

```
    S = input()
    found = re.search(r"^[456]\d{3}(-?)(\d{4}\1){2}\d{4}$",S)
    if found :
        found = re.search(r"(\d)-?(\1-?){3}",S) # NOT have 4 consecutive repe
ated digits
        if found :
            print("Invalid")
        else :
            print("Valid")
    else :
        print("Invalid")
```

Code 21: Word Score

```
def score_words(current_words):
    vowel_chars = 0
    final_score = 0
    list_with_vowels = ["a", "e", "i", "o", "u", "y"]
    for current_word in current_words:
        for current_char in current_word:
            if current_char in list_with_vowels:
                vowel_chars += 1
        if vowel_chars % 2 == 0:
            final_score += 2
            vowel_chars = 0
        else:
            final_score += 1
            vowel_chars = 0
    return final_score
```

```
def main():
    number_of_words = int(input())
    current_words = input().split()
    final_score = score_words(current_words)
    print(final_score)
```

Code 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()):
    if(stream.current % 2 == 1):
```

```
        stream.current = 1
    else:
        stream.current = 0
    for i 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())
```

Code 23: Athlete sort

```
#!/bin/python3

import math
import os
import random
import re
import sys


if __name__ == '__main__':
    nm = input().split()
```

```
n = int(nm[0])

m = int(nm[1])

arr = []



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

k = int(input())
for sorted_array in sorted(arr, key=lambda array: array[k]):
    print(' '.join([str(char) for char in sorted_array]))
```