NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

CSE-860 Artificial Intelligence

ASSIGNMENT NO. 3

Submitted to:
Dr. Yasar Ayaz

Submitted by:
Zahra Batool
451672
M.S Robotics & Intelligent Machine Engineering
Fall 2023

Dated: 01 January 2024

Q: Complete only Medium and Hard challenges of Python from https://www.hackerrank.com/

Challenge 1: Write a Function (**Medium**)

An extra day is added to the calendar almost every four years as February 29, and the day is called a leap day. It corrects the calendar for the fact that our planet takes approximately 365.25 days to orbit the sun. A leap year contains a leap day.

In the Gregorian calendar, three conditions are used to identify leap years:

- The year can be evenly divided by 4, is a leap year, unless:
  - The year can be evenly divided by 100, it is NOT a leap year, unless:
    - The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. Source

Task

Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean `True`, otherwise return `False`.

Note that the code stub provided reads from STDIN and passes arguments to the `is_leap` function. It is only necessary to complete the `is_leap` function.

SOLUTION

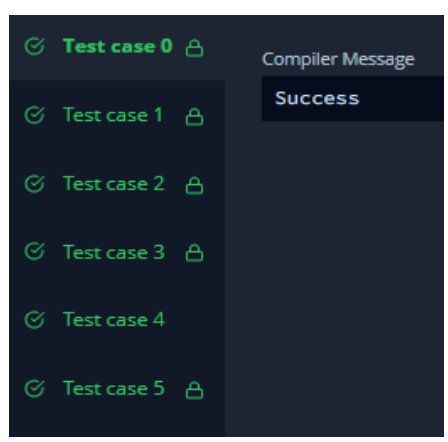
Code:

```
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))
```

Screenshot of Code:

```
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))
```

Result is Successful!



Challenge 2: The Minion Game (**Medium**)

Kevin and Stuart want to play the '**The Minion Game**'.

Game Rules

Both players are given the same string, S.

Both players have to make substrings using the letters of the string S.

Stuart has to make words starting with consonants.

Kevin has to make words starting with vowels.

The game ends when both players have made all possible substrings.

Scoring

A player gets **+1** point for each occurrence of the substring in the string S.

For Example:

String S = BANANA

Kevin's vowel beginning word = ANA

Here, ANA occurs twice in BANANA. Hence, Kevin will get **2** Points.

SOLUTION

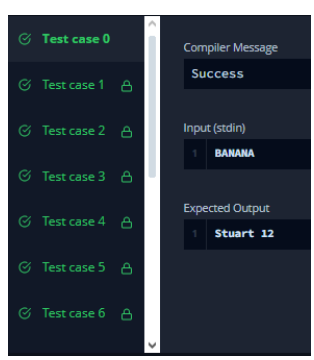
Code:

```
def minion_game(string):
    word=len(string)
    K_score= 0
    S_score= 0
    vowels="AEIOU"

    for i in range(word):
        if string[i] in vowels:
            K_score+= word - i
        else:
            S_score+= word - i

    if K_score > S_score:
        print("Kevin", K_score)
    elif S_score > K_score:
        print("Stuart", S_score)
    else:
        print("Draw")

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

Screenshot of Code:

```
def minion_game(string):
    word=len(string)
    K_score= 0
    S_score= 0
    vowels="AEIOU"

    for i in range(word):
        if string[i] in vowels:
            K_score+= word - i
        else:
            S_score+= word - i

    if K_score > S_score:
        print("Kevin", K_score)
    elif S_score > K_score:
        print("Stuart", S_score)
    else:
        print("Draw")

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

Result is Successful!



Challenge 3: Merge the Tools! (**Medium**)

Consider the following:

- A string, s, of length n where $s = c_0c_1 \ldots c_{n-1}$.
- An integer, k, where k is a factor of n.

We can split s into $\frac{n}{k}$ substrings where each subtring, t_i, consists of a contiguous block of k characters in s. Then, use each t_i to create string u_i such that:

- The characters in u_i are a subsequence of the characters in t_i.
- Any repeat occurrence of a character is removed from the string such that each character in u_i occurs exactly once. In other words, if the character at some index j in t_i occurs at a previous index $< j$ in t_i, then do not include the character in string u_i.

Given s and k, print $\frac{n}{k}$ lines where each line i denotes string u_i.

Function Description

Complete the merge_the_tools function in the editor below.

merge_the_tools has the following parameters:

- string s: the string to analyze
- int k: the size of substrings to analyze

Prints

Print each subsequence on a new line. There will be $\frac{n}{k}$ of them. No return value is expected.

SOLUTION

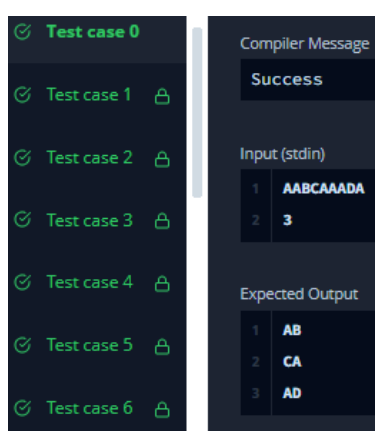
Code:

```
def merge_the_tools(string, k):
    for i in range(0, len(string),k):
        substring= string[i:i+k]
        a = ''

        for char in substring:
            if char not in a:
                a += char

        print(a)

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

Screenshot of Code:

```
def merge_the_tools(string, k):
    for i in range(0, len(string),k):
        substring= string[i:i+k]
        a = ''

        for char in substring:
            if char not in a:
                a += char

        print(a)

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

Result is Successful!



Challenge 4: Time Delta (**Medium**)

When users post an update on social media,such as a URL, image, status update etc., other users in their network are able to view this new post on their news feed. Users can also see exactly when the post was published, i.e, how many hours, minutes or seconds ago.

Since sometimes posts are published and viewed in different time zones, this can be confusing. You are given two timestamps of one such post that a user can see on his newsfeed in the following format:

`Day dd Mon yyyy hh:mm:ss +xxxx`

Here +xxxx represents the time zone. Your task is to print the absolute difference (in seconds) between them.

SOLUTION

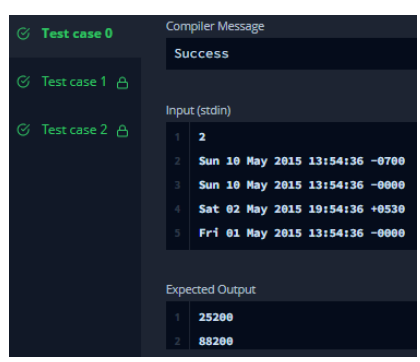
Code:

```
#!/bin/python3
import math
import os
import random
import re
import sys

from datetime import datetime

def time_delta(t1, t2):
    x = '%a %d %b %Y %H:%M:%S %z'
    t1 = datetime.strptime(t1, x)
    t2 = datetime.strptime(t2, x)
    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()
```

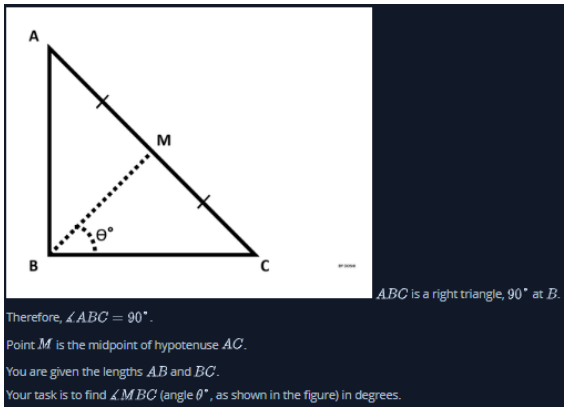
Screenshot of Code:

```
#!/bin/python3

import math
import os
import random
import re
import sys

from datetime import datetime

def time_delta(t1, t2):
    x = '%a %d %b %Y %H:%M:%S %z'
    t1 = datetime.strptime(t1, x)
    t2 = datetime.strptime(t2, x)
    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()
```

Result is Successful!



Challenge 5: Find Angle MBC (**Medium**)



SOLUTION

Code:

```
import math

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

AC = math.sqrt((AB*AB)+(BC*BC))
BM = 0.5*AC
MC = BM

theta_radian = math.acos(BC / (2*MC))
theta_degree = int(round(theta_radian*(180/math.pi)))

print(theta_degree, '\u00B0', sep='')
```

Screenshot of Code:

```
import math

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

AC = math.sqrt((AB*AB)+(BC*BC))
BM = 0.5*AC
MC = BM

theta_radian = math.acos(BC / (2*MC))
theta_degree = int(round(theta_radian * (180 / math.pi)))

print(theta_degree, '\u00B0', sep='')
```

Result is Successful!

Challenge 6: No Idea! (Medium)

There is an array of n integers. There are also 2 **disjoint sets**, A and B, each containing m integers. You like all the integers in set A and dislike all the integers in set B. Your initial happiness is 0. For each i integer in the array, if $i \in A$, you add 1 to your happiness. If $i \in B$, you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.

Note: Since A and B are sets, they have no repeated elements. However, the array might contain duplicate elements.

Constraints

$1 \leq n \leq 10^5$

$1 \leq m \leq 10^5$

$1 \leq Any\ integer\ in\ the\ input \leq 10^9$

SOLUTION

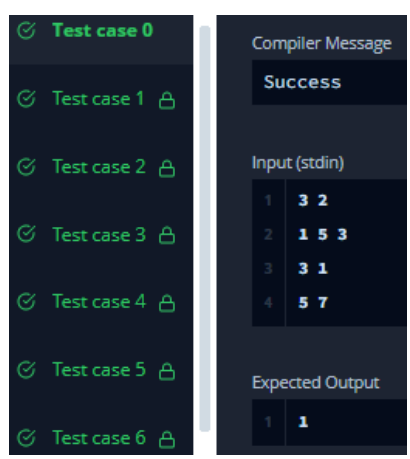
Code:

```
def main():

    happiness = 0

    n, m = map(int, input().strip().split(' '))
    elements_array = list(map(int, input().strip().split(' ')))
    A = set(map(int, input().strip().split(' ')))
    B = set(map(int, input().strip().split(' ')))

    for i in elements_array:
        if i in A:
            happiness += 1
        if i in B:
            happiness -= 1

    print(happiness)

main()
```

Screenshot of Code:

```
def main():

    happiness = 0

    n, m = map(int, input().strip().split(' '))
    elements_array = list(map(int, input().strip().split(' ')))
    A = set(map(int, input().strip().split(' ')))
    B = set(map(int, input().strip().split(' ')))

    for i in elements_array:
        if i in A:
            happiness += 1
        if i in B:
            happiness -= 1

    print(happiness)

main()
```

Result is Successful!



Challenge 7: Word Order (**Medium**)

You are given n words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

Note: Each input line ends with a **"\n"** character.

Constraints:

$1 \leq n \leq 10^5$

The sum of the lengths of all the words do not exceed 10^6

All the words are composed of lowercase English letters only.

SOLUTION

Code:

```
n=int(input())
words=[input() for i in range(n)]
occurences={}

for word in words:
    occurences[word]=0
for word in words:
    occurences[word]+=1

print(len(occurences))
total_occurences=occurences.values()

for i in total_occurences:
    print(i, end=" ")
```

Screenshot of Code:

```
n=int(input())
words=[input() for i in range(n)]
occurences={}

for word in words:
    occurences[word]=0
for word in words:
    occurences[word]+=1

print(len(occurences))
total_occurences=occurences.values()

for i in total_occurences:
    print(i, end=" ")
```

Result is Successful!

Challenge 8: Compress the String! (**Medium**)

In this task, we would like for you to appreciate the usefulness of the groupby() function of itertools . To read more about this function, Check this out .

You are given a string S. Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with $(X,\ c)$ in the string.

For a better understanding of the problem, check the explanation.

SOLUTION

Code:

```
from itertools import groupby

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

Screenshot of Code:

```
from itertools import groupby

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

Result is Successful!

Challenge 9: Company Logo (**Medium**)

A newly opened multinational brand has decided to base their company logo on the three most common characters in the company name. They are now trying out various combinations of company names and logos based on this condition. Given a string *s*, which is the company name in lowercase letters, your task is to find the top three most common characters in the string.

- Print the three most common characters along with their occurrence count.
- Sort in descending order of occurrence count.
- If the occurrence count is the same, sort the characters in alphabetical order.

For example, according to the conditions described above,

GOOGLE would have it's logo with the letters G, O, E.

SOLUTION

Code:

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

if __name__ == '__main__':
    S = input()
    S = sorted(S)
    frequency = Counter(list(S))
    for character, count in frequency.most_common(3):
        print(character, count)
```

Screenshot of Code:

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

if __name__ == '__main__':
    S = input()
    S = sorted(S)
    frequency = Counter(list(S))
    for character, count in frequency.most_common(3):
        print(character, count)
```

Result is Successful!

Challenge 10: Piling Up! (**Medium**)

There is a horizontal row of n cubes. The length of each cube is given. You need to create a new vertical pile of cubes. The new pile should follow these directions: if $cube[i]$ is on top of $cube[j]$ then $sideLength[j] \geq sideLength[i]$.

When stacking the cubes, you can only pick up either the leftmost or the rightmost cube each time. Print `Yes` if it is possible to stack the cubes. Otherwise, print `No`.

SOLUTION

Code:

```
T = int(input())

for _ in range(T):
    n = int(input())
    cubes = list(map(int, input().split()))
    result = "Yes"

    left, right = 0, n - 1

    while left <= right:
        top_cube = max(cubes[left], cubes[right])

        if cubes[left] >= cubes[right]:
            right -= 1
        else:
            left += 1

        if top_cube < cubes[left] or top_cube < cubes[right]:
            result = 'No'
            break

    print(result)
```

Screenshot of Code:

```
T = int(input())

for _ in range(T):
    n = int(input())
    cubes = list(map(int, input().split()))
    result = "Yes"

    left, right = 0, n - 1

    while left <= right:
        top_cube = max(cubes[left], cubes[right])

        if cubes[left] >= cubes[right]:
            right -= 1
        else:
            left += 1

        if top_cube < cubes[left] or top_cube < cubes[right]:
            result = 'No'
            break

    print(result)
```

Result is Successful!

Challenge 11: Triangle Quest 2 (**Medium**)

You are given a positive integer N.

Your task is to print a palindromic triangle of size N.

For example, a palindromic triangle of size 5 is:

```
1
121
12321
1234321
123454321
```

You can't take more than two lines. The first line (a for-statement) is already written for you.

You have to complete the code using exactly one print statement.

SOLUTION

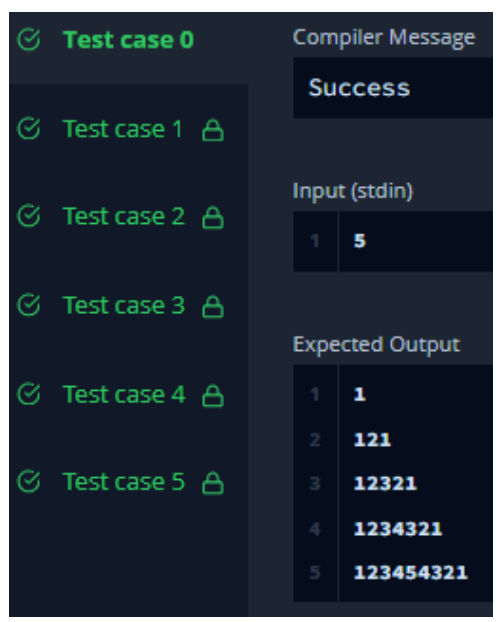
Code:

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

Screenshot of Code:

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

Result is Successful!



Challenge 12: Iterables and Iterators (**Medium**)

The itertools module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an iterator algebra making it possible to construct specialized tools succinctly and efficiently in pure Python.

To read more about the functions in this module, check out their documentation here.

You are given a list of N lowercase English letters. For a given integer K, you can select any K indices (assume 1-based indexing) with a uniform probability from the list.

Find the probability that at least one of the K indices selected will contain the letter: 'a'.

SOLUTION

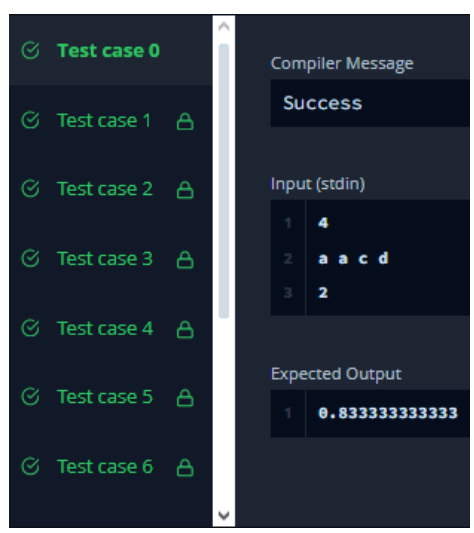
Code:

```
from itertools import combinations
N_length = input()
N_space = input().split()
K = int(input())

data = 0

for i in combinations(N_space, K):
    if 'a' in i:
        data += 1
print(data / len(list(combinations(N_space, K))))
```

Screenshot of Code:

```
from itertools import combinations
N_length = input()
N_space = input().split()
K = int(input())

data = 0

for i in combinations(N_space, K):
    if 'a' in i:
        data += 1
print(data / len(list(combinations(N_space, K))))
```

Result is Successful!



Challenge 13: Triangle Quest (**Medium**)

You are given a positive integer N. Print a numerical triangle of height $N - 1$ like the one below:

```
1
22
333
4444
55555
......
```

Can you do it using only **arithmetic operations, a single for loop and print statement?**

Use no more than two lines. The first line (the for statement) is already written for you. You have to complete the print statement.

Note: Using anything related to strings will give a score of 0.

SOLUTION

Code:

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

Screenshot of Code:

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

Result is Successful!

Challenge 14: Classes: Dealing with Complex Numbers (**Medium**)

For this challenge, you are given two complex numbers, and you have to print the result of their addition, subtraction, multiplication, division and modulus operations.

SOLUTION

Code:

```
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)

    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')
```

Screenshot of Code:

Result is Successful!

Challenge 15: Athlete Sort (**Medium**)

You are given a spreadsheet that contains a list of N athletes and their details (such as age, height, weight and so on). You are required to sort the data based on the Kth attribute and print the final resulting table. Follow the example given below for better understanding.

Rank	Age	Height (in cm)
1	32	190
2	35	175
3	41	188
4	26	195
5	24	176

sort based on k=1 → i.e (age)

Rank	Age	Height (in cm)
5	24	176
4	26	195
1	32	190
2	35	175
3	41	188

Note that K is indexed from 0 to $M-1$, where M is the number of attributes.

Note: If two attributes are the same for different rows, for example, if two atheletes are of the same age, print the row that appeared first in the input.

SOLUTION

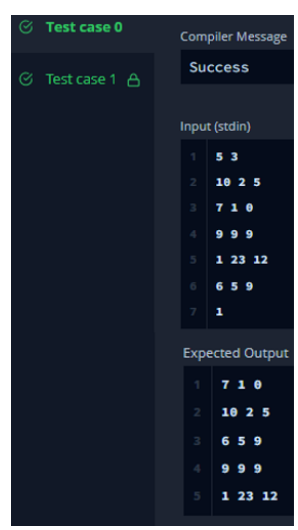
Code:

```
#!/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())

    x=sorted(arr, key=lambda row: row[k])

    for i in range(len(x)):
        for j in range(len(x[i])):
            print(x[i][j], end=' ')
        print()
```

Screenshot of Code:

```
#!/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())

    x=sorted(arr, key=lambda row: row[k])

    for i in range(len(x)):
        for j in range(len(x[i])):
            print(x[i][j], end=' ')
        print()
```

Result is Successful!



Challenge 16: ginortS (**Medium**)

You are given a string S.

S contains alphanumeric characters only.

ginortS

Your task is to sort the string S in the following manner:

- All sorted lowercase letters are ahead of uppercase letters.
- All sorted uppercase letters are ahead of digits.
- All sorted odd digits are ahead of sorted even digits.

SOLUTION

Code:

```
s = input()

string_sorted = sorted(s, key=lambda z: (
    z.isdigit(),
    int(z) % 2 == 0
    if z.isdigit()
    else z.isupper(), z))

print(''.join(string_sorted))
```

Screenshot of Code:

```
s = input()

string_sorted = sorted(s, key=lambda z: (
    z.isdigit(),
    int(z) % 2 == 0
    if z.isdigit()
    else z.isupper(), z))

print(''.join(string_sorted))
```

Result is Successful!

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin)

1 Sorting1234

Expected Output

1 ginortS1324

Challenge 17: Validating Email Addresses with a Filter (Medium)

You are given an integer N followed by N email addresses. Your task is to print a list containing only valid email addresses in lexicographical order.

Valid email addresses must follow these rules:

- It must have the username@websitename.extension format type.
- The username can only contain letters, digits, dashes and underscores $[a-z], [A-Z], [0-9], [_-]$.
- The website name can only have letters and digits $[a-z], [A-Z], [0-9]$.
- The extension can only contain letters $[a-z], [A-Z]$.
- The maximum length of the extension is 3.

SOLUTION

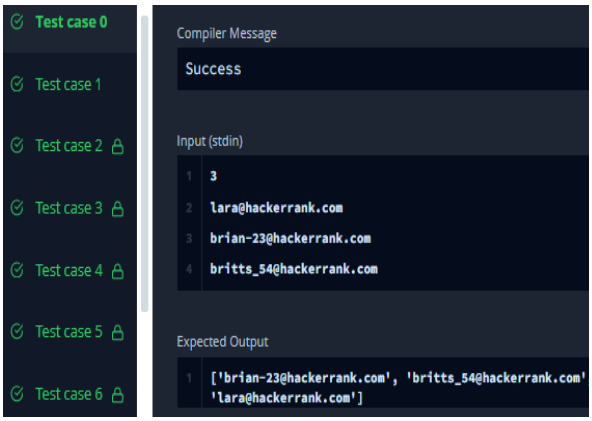
Code:

```
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

    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)
```

Screenshot of Code:

```
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

    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)
```

Result is Successful!



Challenge 18: Reduce Function (Medium)

Given a list of rational numbers,find their product.

Concept

The `reduce()` function applies a function of two arguments cumulatively on a list of objects in succession from left to right to reduce it to one value. Say you have a list, say `[1,2,3]` and you have to find its sum.

```
>>> reduce(lambda x, y : x + y,[1,2,3])
6
```

You can also define an initial value. If it is specified, the function will assume initial value as the value given, and then reduce. It is equivalent to adding the initial value at the beginning of the list. For example:

```
>>> reduce(lambda x, y : x + y, [1,2,3], -3)
3

>>> from fractions import gcd
>>> reduce(gcd, [2,4,8], 3)
1
```

SOLUTION

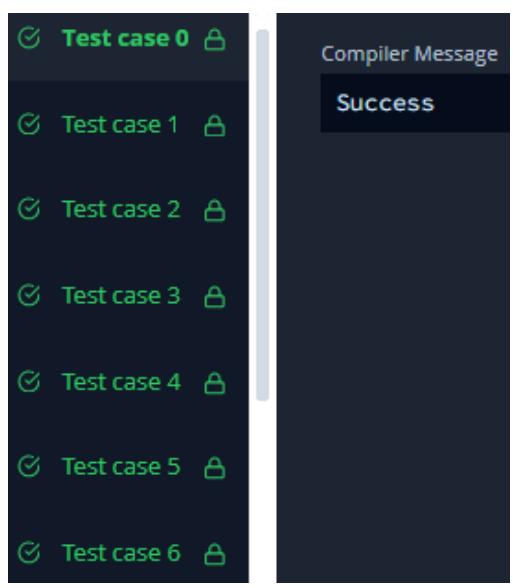
Code:

```
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

if __name__ == '__main__':
    fracs = []
    for _ in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
    print(*result)
```

Screenshot of Code:

```
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

if __name__ == '__main__':
    fracs = []
    for _ in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
    print(*result)
```

Result is Successful!



Challenge 19: Regex Substitution (**Medium**)

The re.sub() tool (sub stands for substitution) evaluates a pattern and, for each valid match, it calls a method (or lambda).

The method is called for all matches and can be used to modify strings in different ways.

The re.sub() method returns the modified string as an output.

Learn more about $re.sub()$.

Task

You are given a text of N lines. The text contains && and || symbols.

Your task is to modify those symbols to the following:

```
&& → and
|| → or
```

Both && and || should have a space " " on both sides.

SOLUTION

Code:

```
import re

n = int(input())

for i in range(n):

    x = input()
    s = re.sub("(?<=\s)&&(?=\s)", "and", x)
    print(re.sub("(?<=\s)\|\|(?=\s)", "or", s))
```

Screenshot of Code:

```
import re

n = int(input())

for i in range(n):

    x = input()
    s = re.sub("(?<=\s)&&(?=\s)", "and", x)
    print(re.sub("(?<=\s)\|\|(?=\s)", "or", s))
```

Result is Successful!

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Test case 7
Test case 8

Compiler Message

Success

Input (stdin)

```
11
a = 1;
b = input();

if a + b > 0 && a - b < 0:
    start()
elif a*b > 10 || a/b < 1:
    stop()
print set(list(a)) | set(list(b))
#Note do not change &&& or ||| or & or |
#Only change those '&&' which have space on both sides.
#Only change those '|| which have space on both sides.
```

Expected Output

```
a = 1;
b = input();

if a + b > 0 and a - b < 0:
    start()
elif a*b > 10 or a/b < 1:
    stop()
print set(list(a)) | set(list(b))
#Note do not change &&& or ||| or & or |
#Only change those '&&' which have space on both sides.
#Only change those '|| which have space on both sides.
```

Challenge 20: Validating Credit Card Numbers (Medium)

You and Fredrick are good friends. Yesterday, Fredrick received N credit cards from **ABCD Bank**. He wants to verify whether his credit card numbers are valid or not. You happen to be great at regex so he is asking for your help!

A valid credit card from **ABCD Bank** has the following characteristics:

- It must start with a 4, 5 or 6.
- It must contain exactly 16 digits.
- It must only consist of digits (0-9).
- It may have digits in groups of 4, separated by one hyphen **"-"**.
- It must **NOT** use any other separator like ' ', '_', etc.
- It must **NOT** have 4 or more consecutive repeated digits.

SOLUTION

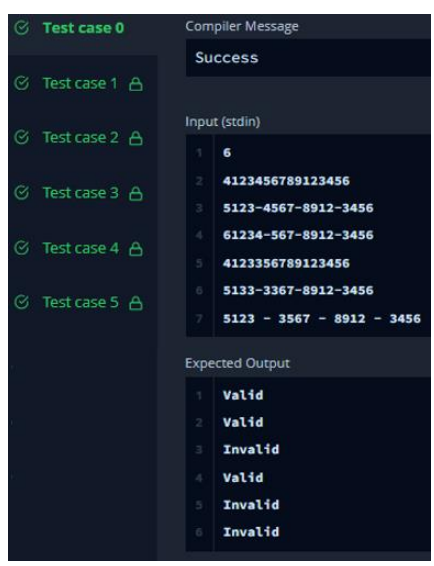
Code:

```
import re

n = int(input())

for test in range(n):
    x = input().strip()

    length_1 = bool(re.match(r"^[456]\d{15}$", x))

    length_2 = bool(re.match(r"^[456]\d{3}\-\d{4}\-\d{4}\-\d{4}$", x))

    x = x.replace("-", "")
    length_3 = bool(re.match(r"(?!.*(\d)(-?\1){3})", x))

    if (length_1 or length_2) and length_3:
        print("Valid")
    else:
        print("Invalid")
```

Screenshot of Code:

```
import re

n = int(input())

for test in range(n):
    x = input().strip()

    length_1 = bool(re.match(r"^[456]\d{15}$", x))

    length_2 = bool(re.match(r"^[456]\d{3}\-\d{4}\-\d{4}\-\d{4}$", x))

    x = x.replace("-", "")
    length_3 = bool(re.match(r"(?!.*(\d)(-?\1){3})", x))

    if (length_1 or length_2) and length_3:
        print("Valid")
    else:
        print("Invalid")
```

Result is Successful!



Challenge 21: Words Score (**Medium**)

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

Consider that vowels in the alphabet are `a, e, i, o, u` and `y`.

Function `score_words` takes a list of lowercase words as an argument and returns a score as follows:

The score of a single word is 2 if the word contains an even number of vowels. Otherwise, the score of this word is 1. The score for the whole list of words is the sum of scores of all words in the list.

Debug the given function `score_words` such that it returns a correct score.

Your function will be tested on several cases by the locked template code.

SOLUTION

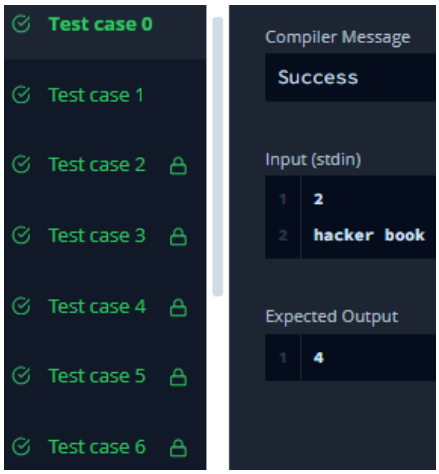
Code:

```
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))
```

Screenshot of 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))
```

Result is Successful!



Challenge 22: Default Arguments (**Medium**)

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

Python supports a useful concept of default argument values. For each keyword argument of a function, we can assign a default value which is going to be used as the value of said argument if the function is called without it. For example, consider the following increment function:

```
def increment_by(n, increment=1):
    return n + increment
```

The functions works like this:

```
>>> increment_by(5, 2)
7
>>> increment_by(4)
5
>>>
```

Debug the given function `print_from_stream` using the default value of one of its arguments.

The function has the following signature:

```
def print_from_stream(n, stream)
```

This function should print the first `n` values returned by `get_next()` method of `stream` object provided as an argument. Each of these values should be printed in a separate line.

Whenever the function is called without the `stream` argument, it should use an instance of `EvenStream` class defined in the code stubs below as the value of `stream`.

Your function will be tested on several cases by the locked template code.

SOLUTION

Code:

```
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=None):
    if stream is None:
        stream = EvenStream()

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

raw = 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())
```

Screenshot of Code:

```
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=None):
    if stream is None:
        stream = EvenStream()

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

raw = 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())
```

Result is Successful!

Challenge 23: Maximize It! (**Hard**)

You are given a function $f(X) = X^2$. You are also given K lists. The ith list consists of N_i elements.

You have to pick one element from each list so that the value from the equation below is maximized:

$$S = (f(X_1) + f(X_2) + \ldots + f(X_k))\%M$$

X_i denotes the element picked from the ith list . Find the maximized value S_{max} obtained.

$\%$ denotes the modulo operator.

Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.

SOLUTION

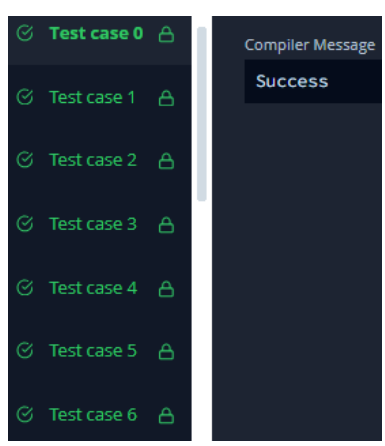
Code:

```
from itertools import product

K, M = map(int, input().split())

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

Max_modulus = float('-inf')

for combination in product(*L):
    S = 0
    for value in combination:
        S += value * value
    Modulus = S % M
    Max_modulus = max(Max_modulus, Modulus)

print(Max_modulus)
```

Screenshot of Code:

```
from itertools import product

K, M = map(int, input().split())

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

Max_modulus = float('-inf')

for combination in product(*L):
    S = 0
    for value in combination:
        S += value * value
    Modulus = S % M
    Max_modulus = max(Max_modulus, Modulus)

print(Max_modulus)
```

Result is Successful!



Challenge 24: Validating Postal Codes (**Hard**)

A valid postal code P have to fullfil both below requirements:

1. P must be a number in the range from 100000 to 999999 inclusive.
2. P must not contain more than one alternating repetitive digit pair.

Alternating repetitive digits are digits which repeat immediately after the next digit. In other words, an alternating repetitive digit pair is formed by two equal digits that have just a single digit between them.

For example:

```
121426 # Here, 1 is an alternating repetitive digit.
523563 # Here, NO digit is an alternating repetitive digit.
552523 # Here, both 2 and 5 are alternating repetitive digits.
```

Your task is to provide two regular expressions **regex_integer_in_range** and **regex_alternating_repetitive_digit_pair**. Where:

regex_integer_in_range should match only integers range from 100000 to 999999 inclusive

regex_alternating_repetitive_digit_pair should find alternating repetitive digits pairs in a given string.

Both these regular expressions will be used by the provided code template to check if the input string P is a valid postal code using the following expression:

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

<u>SOLUTION</u>

Code:

```
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)
```

Screenshot of Code:

```
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)
```

Result is Successful!

Challenge 25: Matrix Script (**Hard**)

Neo has a complex matrix script. The matrix script is a $N \times M$ grid of strings. It consists of alphanumeric characters, spaces and symbols (!,@,#,$,%,&).

Matrix Script

T	s	i
h	%	x
i		#
s	M	
$	a	
#	t	%
i	r	!

Matrix Decoded

```
This$#is% Matrix#  %!
```

To decode the script, Neo needs to read each column and select only the alphanumeric characters and connect them. Neo reads the column from top to bottom and starts reading from the leftmost column.

If there are symbols or spaces between two alphanumeric characters of the decoded script, then Neo replaces them with a single space ' ' for better readability.

Neo feels that there is no need to use '**if**' conditions for decoding.

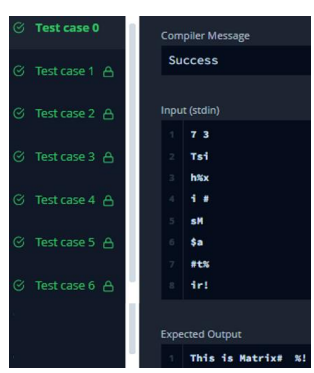
Alphanumeric characters consist of: [A-Z, a-z, and 0-9].

SOLUTION

Code:

```
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 i in range(n):
    matrix_item = input()
    matrix.append(matrix_item)


matrix = list(zip(*matrix))

decoded_string = ""

for words in matrix:
    for char in words:
        decoded_string += char

Result= re.sub(r'(?<=\w)([^\w\d]+)(?=\w)', ' ', decoded_string)
print(Result)
```

Screenshot of Code:

```
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 i in range(n):
    matrix_item = input()
    matrix.append(matrix_item)


matrix = list(zip(*matrix))

decoded_string = ""

for words in matrix:
    for char in words:
        decoded_string += char

Result= re.sub(r'(?<=\w)([^\w\d]+)(?=\w)', ' ', decoded_string)
print(Result)
```

Result is Successful!