



ASSIGNMENT 3

Hacker Rank Problem Solutions

AQIB HABIB MEMON

Reg:450062

Contents

1	Medium Challenges	2
1.1	Athlete Sort	2
1.2	Classes Dealing with Complex Numbers.....	2
1.3	Company Logo.....	3
1.4	Compress the String	4
1.5	Default Arguments	4
1.6	Find Angle MBC.....	5
1.7	ginortS.....	5
1.8	Iterables and Iterators.....	6
1.9	Leap Year	6
1.10	Merge The Tools	7
1.11	Minion Game	8
1.12	No Idea.....	8
1.13	Piling Up	9
1.14	Reduce Function	9
1.15	Regex and Parsing.....	10
1.16	Time Delta.....	10
1.17	Triangle Quest.....	11
1.18	Triangle Quest 2.....	11
1.19	Validating Credit Card Numbers.....	12
1.20	Validating Email Addresses With a Filter	12
1.21	Word Order	13
1.22	Word Score.....	14
2	Hard Challenges.....	15
2.1	Maximize It.....	15
2.2	Validating Postal Codes	15
2.3	Matrix Script	16

1 Medium Challenges

1.1 Athlete Sort

```
import math
import os
import random
import re
import sys
N, M = map(int, input().split())
rows=[]
for _ in range(N):
    rows.append(input())
K = int(input())
for row in sorted(rows, key=lambda row: int(row.split()[K])):
    print(row)
```

Problem

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 K^{th} attribute and print the final resulting table. Follow the example given below for better understanding.

Rank	Age	Height (in cm)	Rank	Age	Height (in cm)
1	32	190	5	24	176
2	35	175	4	26	195
3	41	188	1	32	190
4	26	195	2	35	175
5	24	176	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 athletes are of the same age, print the row that appeared first in the input.

Input Format

The first line contains N and M separated by a space.
The next N lines each contain M elements.
The last line contains K .

Constraints

$1 \leq N, M \leq 1000$
 $0 \leq K < M$

Code Editor

```
K = int(input())
for row in sorted(rows, key=lambda row: int(row.split()[K])):
    print(row)
```

Line: 12 Col: 15

Upload Code as File Test against custom input Run Code Submit Code

You have earned 30.00 points!
15/115 challenges solved. 13%

Congratulations
You solved this challenge. Would you like to challenge your friends?

Next Challenge

1.2 Classes Dealing with Complex Numbers

```
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):
        r = (self.real*no.real)-(self.imaginary*no.imaginary)
        i = (self.real*no.imaginary+no.real*self.imaginary)
        return Complex(r, i)

    def __truediv__(self, no):
        conjugate = Complex(no.real, (-no.imaginary))
        num = self*conjugate
        denom = no*conjugate
        try:
            return Complex((num.real/denom.real), (num.imaginary/denom.real))
        except Exception as e:
            print(e)

    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')

```

Problem

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. The real and imaginary precision part should be correct up to two decimal places.

Input Format

One line of input: The real and imaginary part of a number separated by a space.

Output Format

For two complex numbers C and D , the output should be in the following sequence on separate lines:

- $C + D$
- $C - D$
- $C * D$
- C / D
- $\text{mod}(C)$
- $\text{mod}(D)$

For complex numbers with non-zero real (A) and complex part (B), the output should be in the following format:

$A + Bi$

Replace the plus symbol (+) with a minus symbol (-) when $B < 0$.

For complex numbers with a zero complex part i.e. real numbers, the output should be:

$A + 0.00i$

Submissions

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Editor

```

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())

```

Line: 44 Col: 1

Submissions

You have earned 20.00 points!
14/115 challenges solved. 12%

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) **Next Challenge**

Test case 0 **Compiler Message**

1.3 Company Logo

```

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

```

```

s = input()
s = sorted(s)

```

```

Occurrence= Counter(list(s))
for k, v in Occurrence.most_common(3):
    print(k, v)

```

Output Format

Print the three most common characters along with their occurrence count each on a separate line.

Sort output in descending order of occurrence count.

If the occurrence count is the same, sort the characters in alphabetical order.

Sample Input 0

```
aabbccde
```

Sample Output 0

```

b 3
a 2
c 2

```

Explanation 0

Here, b occurs 3 times. It is printed first.

Both a and c occur 2 times. So, a is printed in the second line and c in the third line because a comes before c in the alphabet.

Note: The string S has at least 3 distinct characters.

Submissions

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Line: 13 Col: 16

Test case 0 **Compiler Message**

Success

Test case 1 **Test case 2** **Test case 3** **Test case 4** **Test case 5**

Input (stdin) **Download**

```

1 aabbccde

```

Expected Output **Download**

```

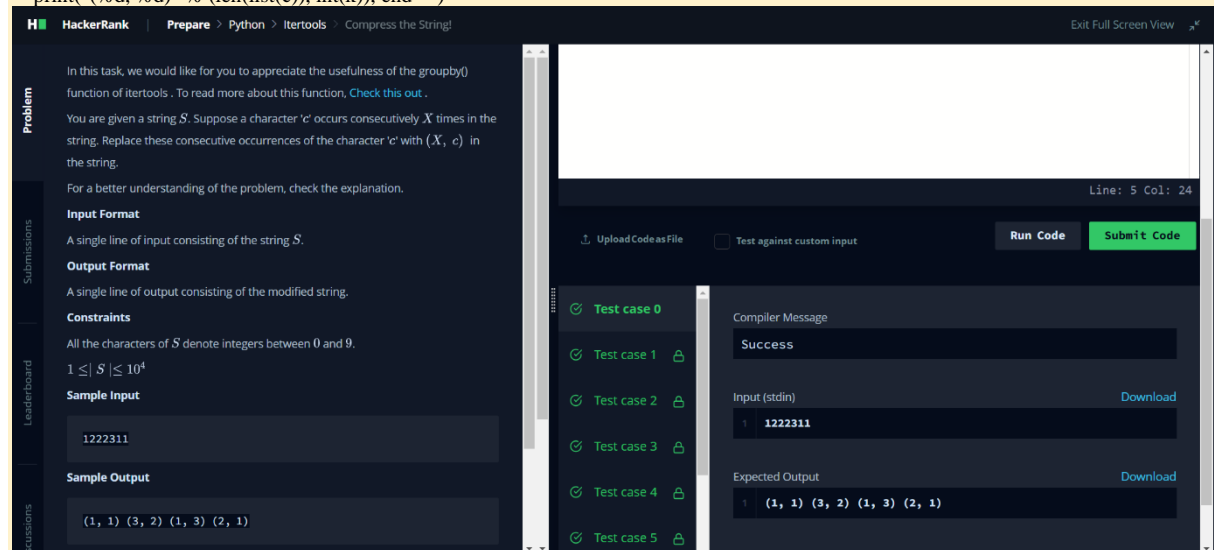
1 b 3
2 a 2
3 c 2

```

1.4 Compress the String

```
from itertools import groupby
str_in=input()
```

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



Problem

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.

Input Format

A single line of input consisting of the string S .

Output Format

A single line of output consisting of the modified string.

Constraints

All the characters of S denote integers between 0 and 9.

$1 \leq |S| \leq 10^4$

Sample Input

```
1222311
```

Sample Output

```
(1, 1) (3, 2) (1, 3) (2, 1)
```

Test cases

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5

Compiler Message

Success

Input (stdin)

```
1 1222311
```

Expected Output

```
1 (1, 1) (3, 2) (1, 3) (2, 1)
```

1.5 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()):
```

```
    stream.__init__()
    for _ 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())
```

HackerRank | Prepare > Python > Debugging > Default Arguments

Problem

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 function 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)
```

Submissions

Leaderboard

Discussions

```
25 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())
```

Line: 25 Col: 1

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Test case 0 **Test case 1** **Test case 2** **Test case 3** **Test case 4** **Test case 5**

Compiler Message

Success

Input (stdin) [Download](#)

```
1 3
2 odd 2
3 even 3
4 odd 5
```

Expected Output [Download](#)

1.6 Find Angle MBC

```
import math
ab=int(input())
bc=int(input())
ca=math.hypot(ab,bc)
mc=ca/2
bca=math.asin(1*ab/ca)
bm=math.sqrt((bc**2+mc**2)-(2*bc*mc*math.cos(bca)))
mbc=math.asin(math.sin(bca)*mc/bm)
print(int(round(math.degrees(mbc), 0)), '\u00B0', sep=)
```

HackerRank | Prepare > Python > Math > Find Angle MBC

Problem

triangle, 90° at B .
Therefore, $\angle ABC = 90^\circ$.
Point M is the midpoint of hypotenuse AC .
You are given the lengths AB and BC .
Your task is to find $\angle MBC$ (angle θ), as shown in the figure) in degrees.

Input Format

The first line contains the length of side AB .
The second line contains the length of side BC .

Constraints

- $0 < AB \leq 100$
- $0 < BC \leq 100$
- Lengths AB and BC are natural numbers.

Output Format

Output $\angle MBC$ in degrees.

Note: Round the angle to the nearest integer.

Examples:

If angle is 56.5000001° , then output **57**.
If angle is 56.5000000° , then output **57**.
If angle is 56.4999999° , then output **56**.
 $0^\circ < \theta^\circ < 90^\circ$

Submissions

Leaderboard

Discussions

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) **Next Challenge**

Test case 0 **Test case 1** **Test case 2** **Test case 3** **Test case 4** **Test case 5**

Compiler Message

Success

Input (stdin) [Download](#)

```
1 10
2 10
```

Expected Output [Download](#)

```
1 45°
```

1.7 ginortS

```
odddigits = []
evendigits=[]
uppercase_letters = []
lowercase_letters = []

S=input()

for char in S:
    if char.isdigit():
        if int(char) % 2 == 0:
            evendigits.append(char)
        elif int(char) % 2 !=0:
            odddigits.append(char)
    elif char.isupper():
        uppercase_letters.append(char)
    elif char.islower():
        lowercase_letters.append(char)
```

```

odddigits=sorted(odddigits)
evendigits=sorted(evendigits)
uppercase_letters=sorted(uppercase_letters, key= lambda x: x.upper())
lowercase_letters=sorted(lowercase_letters, key= lambda x: x.lower())
odddigits="".join(odddigits)
evendigits="".join(evendigits)
lowercase_letters="".join(lowercase_letters)
uppercase_letters="".join(uppercase_letters)

print(lowercase_letters+uppercase_letters+odddigits+evendigits)

```

Problem

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.

Input Format

A single line of input contains the string S .

Constraints

- $0 < \text{len}(S) < 1000$

Output Format

Output the sorted string S .

Sample Input

```
Sorting1234
```

Sample Output

```
ginort51324
```

Code Editor

```

evendigits=sorted(evendigits)
uppercase_letters=sorted(uppercase_letters, key= lambda x: x.upper())
lowercase_letters=sorted(lowercase_letters, key= lambda x: x.lower())
odddigits="".join(odddigits)
evendigits="".join(evendigits)
lowercase_letters="".join(lowercase_letters)
uppercase_letters="".join(uppercase_letters)

print(lowercase_letters+uppercase_letters+odddigits+evendigits)

```

Line: 29 Col: 64

Test Cases

- Test case 0: Success
- Test case 1: Success
- Test case 2: Success
- Test case 3: Success
- Test case 4: Success
- Test case 5: Success

Compiler Message

Success

Input (stdin)

```
Sorting1234
```

Expected Output

```
ginort51324
```

1.8 Iterables and Iterators

from itertools import combinations

```

N=int(input())
M=list(input().split(" "))
L=int(input())
K=list(combinations(M, L))
container= [i for i in K if "a" in i]
print(len(container)/len(K))

```

Problem

indices to be selected.

Output Format

Output a single line consisting of the probability that at least one of the K indices selected contains the letter 'a'.

Note: The answer must be correct up to 3 decimal places.

Constraints

- $1 \leq N \leq 10$
- $1 \leq K \leq N$
- All the letters in the list are lowercase English letters.

Sample Input

```
4
a a c d
2
```

Sample Output

```
0.8333
```

Explanation

All possible unordered tuples of length 2 comprising of indices from 1 to 4 are: (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)

Out of these 6 combinations, 5 of them contain either index 1 or index 2 which are

Code Editor

```

from itertools import combinations

N=int(input())
M=list(input().split(" "))
L=int(input())
K=list(combinations(M, L))
container= [i for i in K if "a" in i]
print(len(container)/len(K))

```

Line: 15 Col: 1

Test Cases

- Test case 0: Success
- Test case 1: Success
- Test case 2: Success
- Test case 3: Success
- Test case 4: Success
- Test case 5: Success

Compiler Message

Success

Input (stdin)

```
4
a a c d
2
```

Expected Output

```
0.833333333333
```

1.9 Leap Year

```

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))

```

1.10 Merge The Tools

```

def merge_the_tools(string, k):
    # your code goes here
    ls = len(string)
    result = []
    for i in range(0, ls, k):
        result.append(string[i:i+k])
    result_nc = []
    seen = set()

    for charline in result:
        output_string = ""
        seen = set()
        for char in charline:
            if char not in seen:
                seen.add(char)
                output_string += char
        result_nc.append(output_string)

    for j in result_nc:
        print(j)

if __name__ == '__main__':
    string, k = input().split()
    merge_the_tools(string, k)

```


Problem

- $1 \leq n \leq 10^4$, where n is the length of s
- $1 \leq k \leq n$
- It is guaranteed that n is a multiple of k .

Sample Input

```
STDIN      Function
-----
AABCAAADA  s = 'AABCAAADA'
3          k = 3
```

Sample Output

```
AB
CA
AD
```

Explanation

Split s into $\frac{n}{k} = \frac{9}{3} = 3$ equal parts of length $k = 3$. Convert each t_i to u_i by removing any subsequent occurrences of non-distinct characters in t_i :

- $t_0 = "AAB" \rightarrow u_0 = "AB"$
- $t_1 = "CAA" \rightarrow u_1 = "CA"$
- $t_2 = "ADA" \rightarrow u_2 = "AD"$

Print each u_i on a new line.

Compiler Message

Success

Input (stdin)

```
1 AABCAAADA
2 3
```

Expected Output

```
1 AB
2 CA
3 AD
```

Test cases: Test case 10, 11, 12, 13, 14, 15, 16 (all passing)

1.11 Minion Game

```
def minion_game(string):
    vowels = 'aeiouAEIOU'
    Kevin_Score = int(0)
    Stuart_Score = int(0)
    l = int(len(string))

    for i in range(l):
        if string[i] in vowels:
            Kevin_Score += l - i
        else:
            Stuart_Score += l - i
    if Stuart_Score > Kevin_Score:
        print("Stuart", Stuart_Score)
    elif Stuart_Score == Kevin_Score:
        print("Draw")
    else:
        print("Kevin", Kevin_Score)
```

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

minion_game has the following parameters:

- string $string$: the string to analyze

Prints

- string: the winner's name and score, separated by a space on one line, or *Draw* if there is no winner

Input Format

A single line of input containing the string S .

Note: The string S will contain only uppercase letters: $[A - Z]$.

Constraints

$0 < len(S) \leq 10^6$

Sample Input

```
BANANA
```

Sample Output

```
Stuart 12
```

Note:

Vowels are only defined as *AEIOU*. In this problem, *Y* is not considered a vowel.

Compiler Message

Success

Input (stdin)

```
1 BANANA
```

Expected Output

```
1 Stuart 12
```

Test cases: Test case 0, 1, 2, 3, 4, 5, 6 (all passing)

1.12 No Idea

```
happiness = 0
```

```
n, m = map(int, input().split(' '))
arr = list(map(int, input().split(' ')))
good_bhαι = set(map(int, input().split(' ')))
bad_bhαι = set(map(int, input().split(' ')))
```

```
for i in arr:
    if i in good_bhαι:
        happiness += 1
    elif i in bad_bhαι:
        happiness -= 1
print(happiness)
```

1.13 Piling Up

```
Cube = int(input())
ANS = []
for _ in range(Cube):
    n = int(input())
    cube_list = list(map(int, input().split()))

    for _ in range(n-1):
        if cube_list[0] >= cube_list[len(cube_list)-1]:
            a = cube_list[0]
            cube_list.pop(0)
        elif cube_list[0] < cube_list[len(cube_list)-1]:
            a = cube_list[len(cube_list)-1]
            cube_list.pop(len(cube_list)-1)
        else:
            pass

    if len(cube_list) == 1:
        ANS.append("Yes")

    if ((cube_list[0] > a) or (cube_list[len(cube_list)-1] > a)):
        ANS.append("No")
        break
```

```
print("\n".join(ANS))
```

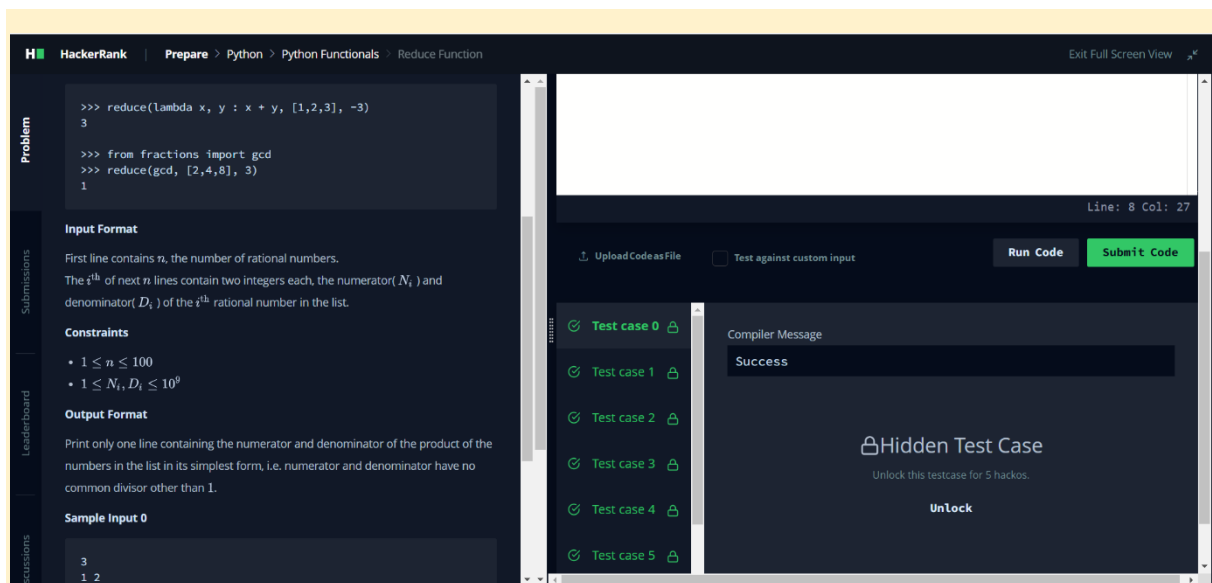
The screenshot shows the HackerRank interface for the 'Piling Up' problem. The problem description states that blocks of size 1, 2, 3, 7, and 8 are available, and the goal is to stack them from right to left. The input format specifies the number of test cases T and the number of cubes n for each test case. Constraints include $1 \leq T \leq 5$, $1 \leq n \leq 10^5$, and $1 \leq \text{sideLength} < 2^{31}$. The sample input shows 2 test cases: the first with $T=2$ and the second with $T=2$. The code editor on the right contains the solution code, and the test cases section shows all five test cases passing successfully.

1.14 Reduce Function

```
from fractions import Fraction
from functools import reduce
```

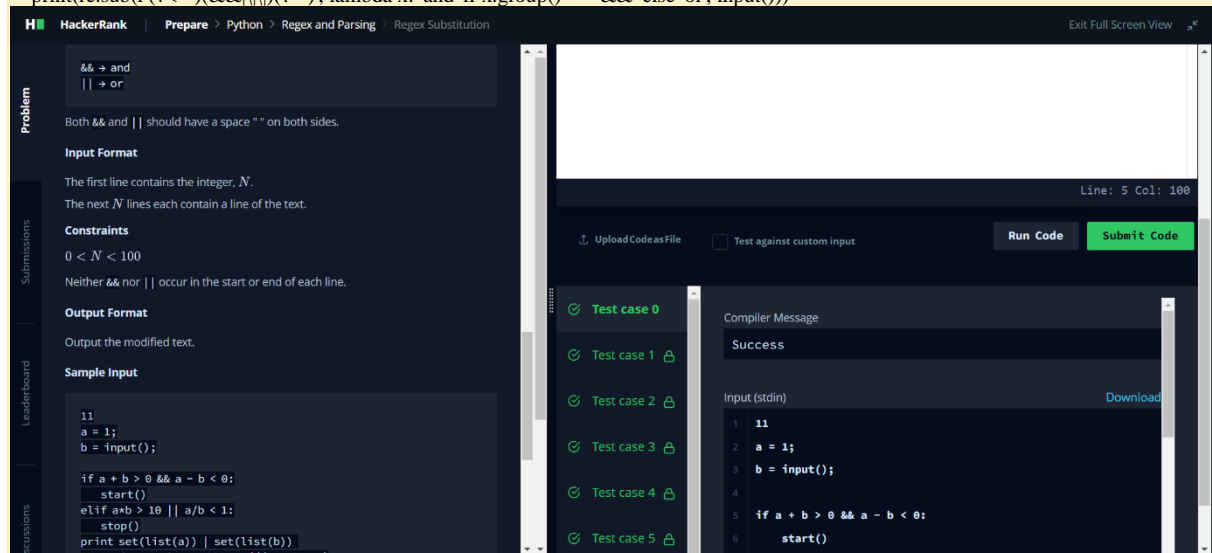
```
def product(fracs):
    t = reduce(lambda x, y: x * y, fracs)
    return t.numerator, t.denominator
```

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



1.15 Regex and Parsing

```
import re
for _ in range(int(input())):
    print(re.sub(r'(?=>)(&&|\\|)(?=<)', lambda x: 'and' if x.group() == '&&' else 'or', input()))
```



1.16 Time Delta

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

# Complete the time_delta function below.
from datetime import datetime
def time_delta(t1, t2):
    time_format = '%a %d %b %Y %H:%M:%S %z'
    t1 = datetime.strptime(t1, time_format)
    t2 = datetime.strptime(t2, time_format)
    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()

```

The screenshot shows the HackerRank interface for the 'Time Delta' challenge. On the left, the 'Problem' tab is active, displaying the problem description: '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.' It also includes input and output formats and constraints. The right panel shows the 'Test case 0' tab with a 'Success' compiler message and a table of test cases. A green 'Congratulations' banner at the top right indicates the user has solved the challenge.

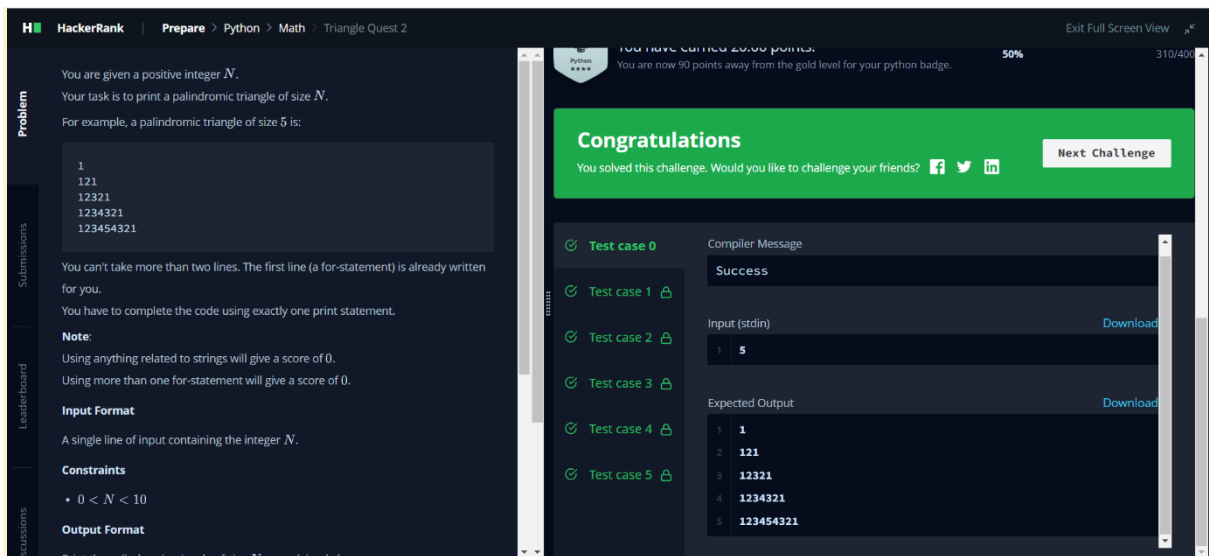
1.17 Triangle Quest

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

The screenshot shows the HackerRank interface for the 'Triangle Quest' challenge. The left panel displays the problem description: '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.' It also includes input and output formats and constraints. The right panel shows the 'Run Code' button and a green 'Congratulations' banner indicating the user has solved the challenge.

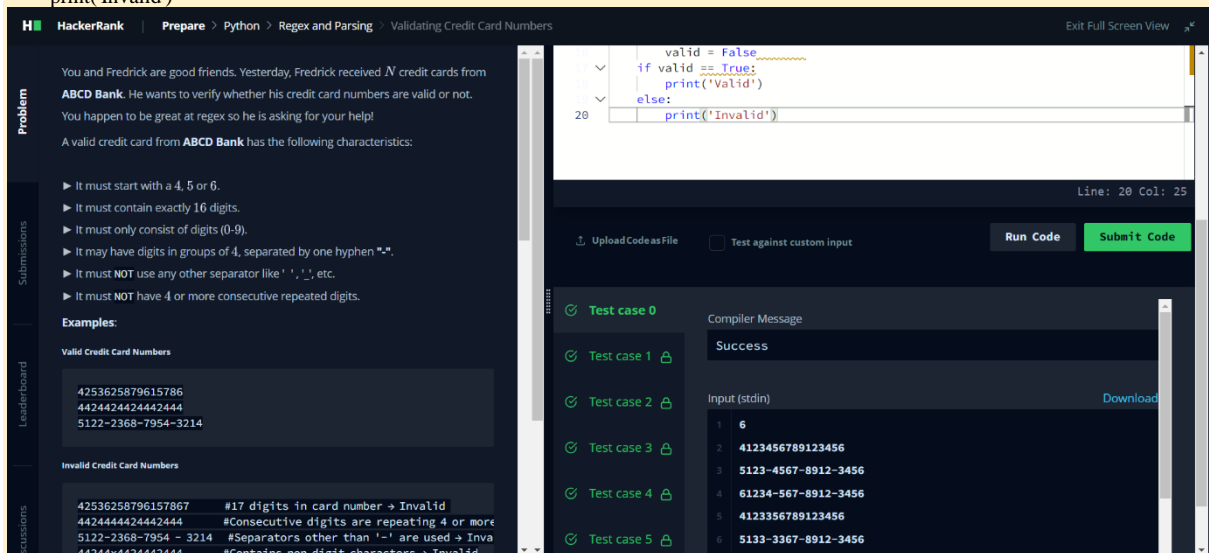
1.18 Triangle Quest 2

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)//9)**2)



1.19 Validating Credit Card Numbers

```
import re
n = int(input())
for t 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\11', credit_removed_hiphen))
    if length_16 == True or length_19 == True:
        if consecutive == True:
            valid = False
        else:
            valid = False
    if valid == True:
        print('Valid')
    else:
        print('Invalid')
```



1.20 Validating Email Addresses With a Filter

```
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
    elif extension.isalpha() is False:
        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)

```

HackerRank | Prepare > Python > Python Functionals | Validating Email Addresses With a Filter

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.

Concept

A filter takes a function returning True or False and applies it to a sequence, returning a list of only those members of the sequence where the function returned True. A Lambda function can be used with filters.

Let's say you have to make a list of the squares of integers from 0 to 9 (both included).

```
>> l = list(range(10))
>> l = list(map(lambda x:x*x, l))
```

Now, you only require those elements that are greater than 10 but less than 80.

Test case 0 ✓

Test case 1 ✓

Test case 2 ✓

Test case 3 ✓

Test case 4 ✓

Test case 5 ✓

Test case 6 ✓

Compiler Message

Success

Input (stdin)

```
3
lara@hackerrank.com
brian-23@hackerrank.com
britts_54@hackerrank.com
```

Expected Output

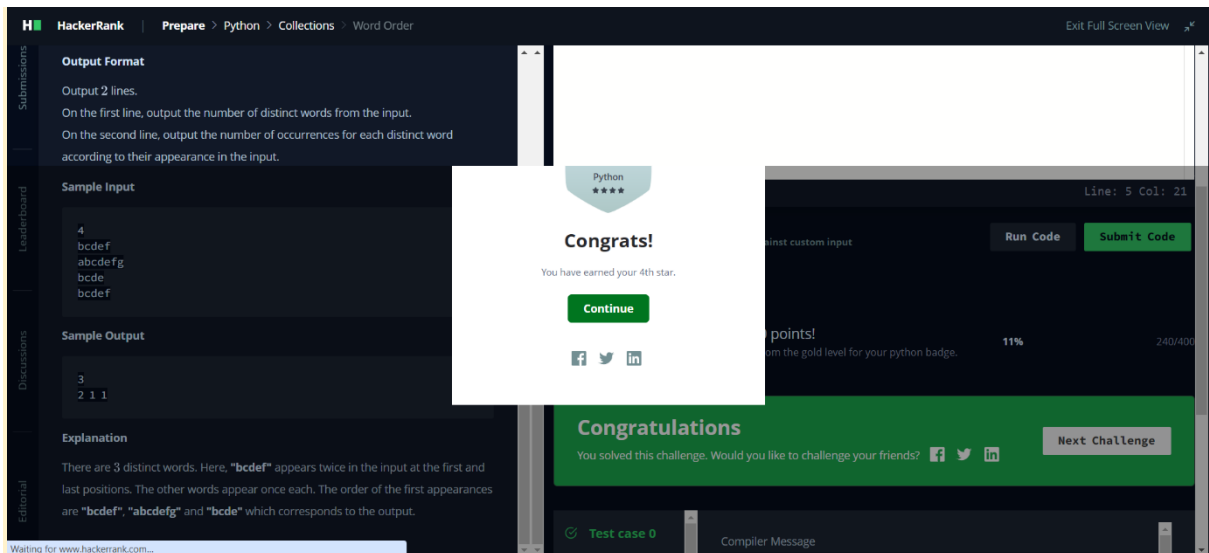
```
['brian-23@hackerrank.com', 'britts_54@hackerrank.com', 'lara@hackerrank.com']
```

1.21 Word Order

```

from collections import Counter
N = int(input())
words = []
for i in range(N):
    words.append(input().strip())
count = Counter(words)
print(len(count))
print(*count.values())

```



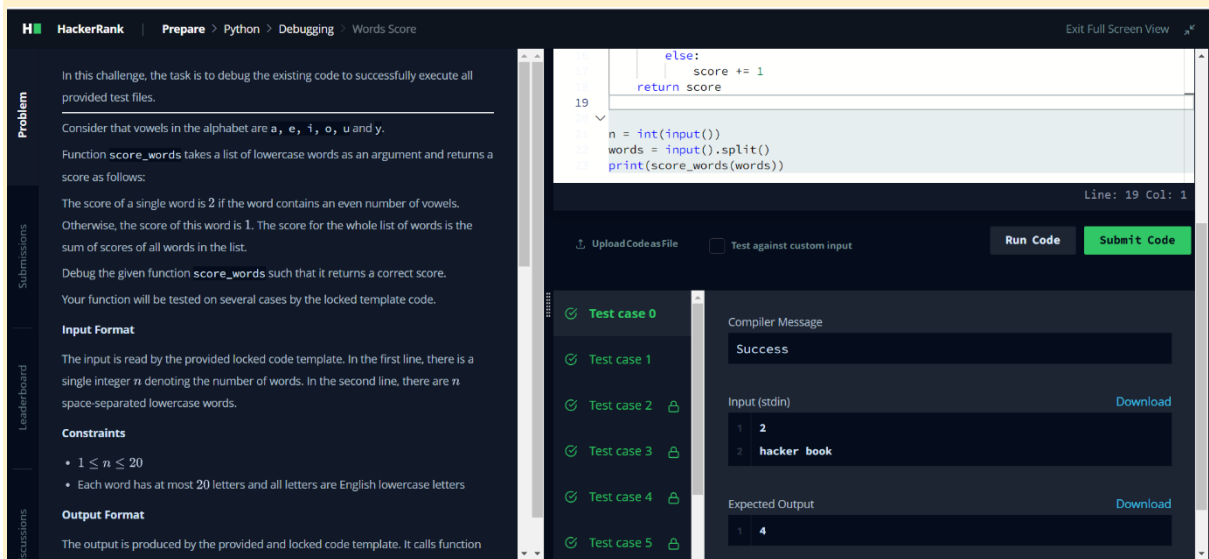
1.22 Word Score

```
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']
```

```
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))
```



2 Hard Challenges

2.1 Maximize It

```
import itertools

NUMBER_OF_LISTS, MODULUS = map(int, input().split())
LISTS_OF_LISTS = []

for i in range(0, NUMBER_OF_LISTS):
    new_list = list(map(int, input().split()))
    del new_list[0]
    LISTS_OF_LISTS.append(new_list)

def squared(element):
    return element**2

COMBS = list(itertools.product(*LISTS_OF_LISTS))
RESULTS = []

for i in COMBS:
    result1 = sum(map(squared, [a for a in i]))
    result2 = result1 % MODULUS
    RESULTS.append(result2)

print(max(RESULTS))
```

Problem

You are given a function $f(X) = X^2$. You are also given K lists. The i^{th} 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:

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

X_i denotes the element picked from the i^{th} 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.

Input Format

The first line contains 2 space separated integers K and M .

The next K lines each contains an integer N_i , denoting the number of elements in the i^{th} list, followed by N_i space separated integers denoting the elements in the list.

Constraints

- $1 \leq K \leq 7$
- $1 \leq M \leq 1000$

Code

```
RESULTS = []
for i in COMBS:
    result1 = sum(map(squared, [a for a in i]))
    result2 = result1 % MODULUS
    RESULTS.append(result2)
print(max(RESULTS))
```

Line: 21 Col: 28

Upload Code as File ☐ Test against custom input **Run Code** **Submit Code**

Test Cases

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5

Compiler Message

Success

Input (stdin)

```
7 867
7 6429964 4173738 9941618 2744666 5392018 5813128 9452095
7 6517823 4135421 6418713 9924958 9370532 7940650 2027017
7 1506500 3460933 1550284 3679489 4538773 5216621 5645660
7 7443563 5181142 8804416 8726696 5358847 7155276 4433125
7 2230555 3920370 7851992 1176871 610460 309961 3921536
```

2.2 Validating Postal Codes

```
regex_integer_in_range = r"[1-9][\d]{5}$" # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)" # Do not delete 'r'.
import re
P = input()
print (bool(re.match(regex_integer_in_range, P))
and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```


HackerRank | Prepare > Python > Regex and Parsing > Validating Postal Codes

Exit Full Screen View

Problem

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

- P must be a number in the range from 100000 to 999999 inclusive.
- 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))
```

Submissions

Leaderboard

Discussions

Line: 3 Col: 1

Upload Code as File Test against custom input

Run Code Submit Code

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin)

110000

Download

Expected Output

False

Download

2.3 Matrix Script

```
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 = []
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)
```

HackerRank | Prepare > Python > Regex and Parsing > Matrix Script

Exit Full Screen View

Problem

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
s  %  #
$  a  %
#  t  %
i  r  i
```

Matrix Decoded

```
This is Matrix0 %!
```

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]`.

Input Format

The first line contains space-separated integers: N (rows) and M (columns) respectively.

Submissions

Leaderboard

Discussions

Line: 19 Col: 28

Upload Code as File Test against custom input

Run Code Submit Code

You have earned 100.00 points!

25/115 challenges solved.

22%

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Compiler Message