

Name: Muhammad Owais

Registration No. 451805

Assignment 3

Artificial Intelligence

Medium Level Challenges:

1. Write a Function:

The screenshot shows the HackerRank challenge interface for the 'Write a Function' problem. The left sidebar includes navigation links for Problem, Submissions, Leaderboard, and Constraints. The main content area contains the challenge details and a code editor.

Challenge Details:

- 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:
 - o 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.

Input Format

Read `year`, the year to test.

Constraints

The code editor shows the following Python 3 code:

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

    # Check the conditions for a leap year
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                leap = True
            else:
                leap = False
        else:
            leap = True

    return leap

> year = int(input())...
```

Below the code editor are buttons for `Upload Code as File`, `Test against custom input`, `Run Code`, and `Submit Code`.

Result:

A green 'Congratulations' banner appears, stating: "You solved this challenge. Would you like to challenge your friends?". It includes social sharing icons for Facebook, Twitter, and LinkedIn, and a 'Next Challenge' button.

The results section shows the following test cases and their outcomes:

Test Case	Input (stdin)	Expected Output	Compiler Message
Test case 0	1 2000	1 True	Success
Test case 1	1 1900	1 False	
Test case 2	1 2100	1 False	
Test case 3	1 2200	1 False	
Test case 4	1 2300	1 False	
Test case 5	1 2500	1 False	

At the bottom right, there is a link to a 'Hidden Test Case'.

2. The Minion Game:

The screenshot shows the HackerRank interface for the 'The Minion Game' challenge. On the left, the problem statement details the game rules: both players (Kevin and Stuart) start with the same string S . They take turns picking substrings from S , starting with consonants for Kevin and vowels for Stuart. Points are awarded for each occurrence of the chosen substring in the original string. For example, in the string 'BANANA', 'ANA' is a vowel beginning word and thus worth points for Stuart.

The right side shows a code editor with Python 3 selected. The code implements the logic described in the problem statement. It defines vowels as 'AEIOU', initializes scores for both players to 0, and calculates the length of the input string. It then iterates through each character of the string, incrementing the appropriate player's score based on whether it's a vowel or consonant. Finally, it prints the winner or 'Draw' if the scores are equal.

```
vowels = "AEIOU"
player1_score = 0
player2_score = 0
str_len = len(string)

for i in range(str_len):
    if string[i] in vowels:
        player2_score += (str_len - i)
    else:
        player1_score += (str_len - i)

if player1_score > player2_score:
    print("Stuart {player1_score}")
elif player1_score < player2_score:
    print(f"Kevin {player2_score}")
else:
    print("Draw")
```

Below the code editor are buttons for 'UploadCodeasFile', 'Test against custom input', 'Run Code', and 'Submit Code'.

The screenshot shows the 'Congratulations' message after solving the challenge. It includes social sharing options (Facebook, Twitter, LinkedIn) and a 'Next Challenge' button.

The right panel displays the test results for the challenge. It shows 'Compiler Message' indicating 'Success', 'Input (stdin)' showing the input '1 BANANA', and 'Expected Output' showing the output '1 Stuart 12'. There are also sections for 'Test case 0' through 'Test case 6'.

Test Case	Input (stdin)	Expected Output
Test case 0	1 BANANA	1 Stuart 12
Test case 1		
Test case 2		
Test case 3		
Test case 4		
Test case 5		
Test case 6		

3. Merge the Tools:

The screenshot shows the HackerRank platform displaying the 'Merge the Tools' challenge. The interface includes a navigation bar at the top, a sidebar on the left with tabs for 'Problem', 'Submissions', 'Leaderboard', and 'Editor'. The main content area contains the problem statement, example input, and output, followed by a code editor and a results section.

Problem Statement:

Consider the following:

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

We can split s into $\frac{n}{k}$ substrings where each substring, 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 .

Example:

$s = \text{'AAABCADDE'}$
 $k = 3$

There are three substrings of length 3 to consider: 'AAA', 'BCA' and 'DDE'. The first substring is all 'A' characters, so $u_1 = \text{'A'}$. The second substring has all distinct characters, so $u_2 = \text{'BCA'}$. The third substring has 2 different characters, so $u_3 = \text{'DE'}$. Note that a subsequence maintains the original order of characters encountered. The order of characters in each

Code Editor:

```
1 def merge_the_tools(string, k):
n = len(string)
for i in range(0, n, k):
    substring = string[i:i+k]
    unique_chars = ""
    for char in substring:
        if char not in unique_chars:
            unique_chars += char
    print(unique_chars)

> if __name__ == '__main__':

```

Line: 1 Col: 1

Test against custom input

Results Section:

Congratulations

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

[Next Challenge](#)

Compiler Message: Success

Input (stdin)

```
1 AABCAAADA
2 3
```

Expected Output

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

4. Time Delta:

The screenshot shows the HackerRank platform interface for a 'Time Delta' problem. The left side features a sidebar with navigation links: Problem, Submissions, and Leaderboard. The main content area displays the problem statement, input/output formats, constraints, and a code editor.

Problem Statement: 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.

Input Format: Day dd Mon yyyy hh:mm:ss +xxxx

Output Format: Print the absolute difference ($t_1 - t_2$) in seconds.

Constraints:

- Input contains only valid timestamps
- year ≤ 3000 .

Code Editor (Top):

```
#!/bin/python3

import math
import os
import random
import re
import sys
from datetime import datetime
```

Code Editor (Bottom):

```
# Complete the time_delta function below.
def time_delta(t1, t2):
    # Define the format of the timestamp
    format_str = "%a %d %b %Y %H:%M:%S %z"

    # Parse the timestamps using the format
    dt1 = datetime.strptime(t1, format_str)
    dt2 = datetime.strptime(t2, format_str)

    # Calculate the absolute difference in seconds
    delta_seconds = int(abs((dt1 - dt2).total_seconds()))
```

Buttons at the bottom of the code editor: UploadCodeasFile, Test against custom input, Run Code, Submit Code.

Line 39 Col: 1

Code Editor (Bottom):

```
# Calculate the absolute difference in seconds
delta_seconds = int(abs((dt1 - dt2).total_seconds()))

return str(delta_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()
```

Buttons at the bottom of the code editor: UploadCodeasFile, Test against custom input, Run Code, Submit Code.

Line 39 Col: 1

HackerRank | Prepare > Python > Date and Time > Time Delta

Problem

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.

Input Format

The first line contains T , the number of testcases.
Each testcase contains 2 lines, representing time t_1 and time t_2 .

Constraints

- Input contains only valid timestamps
- $year \leq 3000$.

Output Format

Print the absolute difference ($t_1 - t_2$) in seconds.

Congratulations
You solved this challenge. Would you like to challenge your friends?
[Next Challenge](#)

Test case 0 Compiler Message Success
Input (stdin) Download
1 2
2 Sun 10 May 2015 13:54:36 -0700
3 Sun 10 May 2015 13:54:36 -0000
4 Sat 02 May 2015 19:54:36 +0530
5 Fri 01 May 2015 13:54:36 -0000

Expected Output Download
1 25200

5. Find Angle MBC:

HackerRank | Prepare > Python > Math > Find Angle MBC

Problem

ABC is a right 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

import math
side_AB = float(input())
side_BC = float(input())
angle_theta = math.degrees(math.atan2(side_AB, side_BC))
rounded_angle = round(angle_theta)
print("{}{}".format(rounded_angle, chr(176)))

Line: 11 Col: 1

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

HackerRank | Prepare > Python > Math > Find Angle MBC

Problem

ABC is a right 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

Congratulations
You solved this challenge. Would you like to challenge your friends?
[Facebook](#) [Twitter](#) [LinkedIn](#)

Test case 0 Compiler Message Success
Input (stdin) Download
1 10
2 10
Expected Output Download
1 45°

6. No Idea:

HackerRank | Prepare > Python > Sets > No Idea

Problem

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

$1 \leq n \leq 10^5$
 $1 \leq m \leq 10^5$
 $1 \leq \text{Any integer in the input} \leq 10^9$

Input Format

The first line contains integers n and m separated by a space.
The second line contains n integers, the elements of the array.
The third and fourth lines contain m integers, A and B , respectively.

Output Format

Output a single integer, your total happiness.

Code Editor

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Read input values
n, m = map(int, input().split())
array = list(map(int, input().split()))
set_a = set(map(int, input().split()))
set_b = set(map(int, input().split()))

# Initialize happiness
happiness = 0

# Calculate happiness
for num in array:
    if num in set_a:
        happiness += 1
    elif num in set_b:
        happiness -= 1

# Print the final happiness
print(happiness)
```

Line: 14 Col: 23

HackerRank | Prepare > Python > Sets > No Ideal

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

$1 \leq n \leq 10^5$
 $1 \leq m \leq 10^5$
 $1 \leq \text{Any integer in the input} \leq 10^9$

Input Format

The first line contains integers n and m separated by a space.
The second line contains n integers, the elements of the array.
The third and fourth lines contain m integers, A and B , respectively.

Output Format

Output a single integer, your total happiness.

Waiting for www.hackerrank.com...

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

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

Loading testcase ...

7. Word Order:

HackerRank | Prepare > Python > Collections > Word Order

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.

Input Format

The first line contains the integer, n .
The next n lines each contain a word.

Output Format

Output 2 lines.
On the first line, output the number of distinct words from the input.
On the second line, output the number of occurrences for each distinct word according to their appearance in the input.

Sample Input

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

# Read input
n = int(input())
words = []

# Collect words in input order
for _ in range(n):
    word = input().strip()
    words.append(word)

# Count occurrences using OrderedDict to maintain input order
word_count = OrderedDict()
for word in words:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1

# Output the result
print(len(word_count))
print(' '.join(str(word_count[word]) for word in word_count))
```

Line: 24 Col: 1

UploadCodeAsFile Test against custom input Run Code Submit Code

The screenshot shows the HackerRank platform interface for the 'Word Order' challenge. On the left, there's a sidebar with 'Problem' selected, followed by 'Submissions' and 'Leaderboard'. The main area contains problem details: a note about input words, constraints ($1 \leq n \leq 10^5$), and a sample input/output. Below this are sections for 'Input Format' (first line integer n , next n lines words) and 'Output Format' (two lines: count of distinct words, then occurrences). A 'Sample Input' section is also present. On the right, a green 'Congratulations' banner says 'You solved this challenge. Would you like to challenge your friends?'. It includes social sharing icons (Facebook, Twitter, LinkedIn) and a 'Next Challenge' button. Below the banner, a 'Compiler Message' section shows 'Success'. Under 'Input (stdin)', the sample input is shown: 4, bcdef, abcdefg, bcde, bcdef. Under 'Expected Output', the result is shown: 3.

8. Compress the String:

```
from itertools import groupby

def compress_string(s):
    compressed_string = []
    for key, group in groupby(s):
        count = len(list(group))
        compressed_string.append((int(key), count))

    result = ' '.join([f"({count}, {char})" for char, count in compressed_string])
    return result

# Read input
s = input().strip()

# Output the modified string
result = compress_string(s)
print(result)
```

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

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

[Next Challenge](#)

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) Download
1 1222311

Expected Output Download
1 (1, 1) (3, 2) (1, 3) (2, 1)

9. Company Logo:

```
#!/bin/python3

from collections import Counter

def top_three_characters(s):
    # Count occurrences of each character
    char_count = Counter(s)

    # Sort by occurrence count (descending) and then by character
    # (ascending)
    sorted_chars = sorted(char_count.items(), key=lambda x: (-x[1], x[0]))

    # Print the top three characters and their occurrence count
    for char, count in sorted_chars[:3]:
        print(f"{char} {count}")

if __name__ == '__main__':
    s = input().strip()
    top_three_characters(s)
```

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 its logo with the letters **G, O, L**.

Input Format

A single line of input containing the string S .

Constraints

- $3 < \text{len}(S) \leq 10^4$
- S has at least 3 distinct characters

Output Format

Congratulations

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

Compiler Message: Success

Test case 0: Input (stdin) 1 aabbccdde

Test case 0: Expected Output 1 b 3
2 a 2
3 c 2

Test case 1: Success

Test case 2: Success

Test case 3: Success

Test case 4: Success

Test case 5: Success

Next Challenge

10. Piling Up:

```

def can_stack_cubes(test_cases):
    for cubes in test_cases:
        left, right = 0, len(cubes) - 1
        top_cube = float('inf')

        while left <= right:
            if cubes[left] >= cubes[right] and cubes[left] <= top_
                cube:
                top_cube = cubes[left]
                left += 1
            elif cubes[right] >= cubes[left] and cubes[right] <= t
                op_cube:
                top_cube = cubes[right]
                right -= 1
            else:
                print("No")
                break
        else:
            print("Yes")

if __name__ == '__main__':
    t = int(input().strip()) # Number of test cases
    test_cases = []

    for _ in range(t):
        _ = int(input().strip()) # Not needed
    
```

```

cubes = list(map(int, input().split()))
test_cases.append(cubes)

can_stack_cubes(test_cases)

```

The screenshot shows the HackerRank interface for the 'Piling Up!' challenge. On the left, the problem statement describes a horizontal row of cubes where each cube's length is given. The task is to stack these cubes vertically while maintaining the original order from left to right. If a cube at index i has a side length of s_i , it can only be placed on top of a cube at index j if $s_j \geq s_i$. The code provided uses a list comprehension to map each input string to an integer, then appends the resulting list to a variable named 'test_cases'. A function 'can_stack_cubes' is then called with 'test_cases' as its argument. The right side of the screen displays a green 'Congratulations' banner with social sharing options. Below it, the 'Test case 0' section shows a 'Success' compiler message. The 'Input (stdin)' section contains five lines of integers: 1, 2, 6, 4 3 2 1 3 4, 3, and 1 3 2. The 'Expected Output' section shows a single line: 1 Yes.

11. Triangle Quest 2:

```

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

```

The screenshot shows the HackerRank interface for the 'Triangle Quest 2' challenge. The problem statement asks to print a palindromic triangle of size N . An example for size 5 is shown as:
`1
121
12321
1234321
123454321`
It notes that only one print statement is allowed. The code provided uses a for-loop to generate the triangle. The right side of the screen displays a green 'Congratulations' banner with social sharing options. Below it, the 'Test case 0' section shows a 'Success' compiler message. The 'Input (stdin)' section contains a single line: 5. The 'Expected Output' section shows a five-line palindromic triangle:
`1
121
12321
1234321
123454321`

12. Iterables and Iterators

```
from itertools import combinations

def calculate_probability(n, letters, k):
    total_combinations = list(combinations(letters, k))
    combinations_without_a = [comb for comb in total_combinations
        if 'a' not in comb]

    probability_no_a = len(combinations_without_a) / len(total_combinations)
    probability_at_least_one_a = 1 - probability_no_a

    return round(probability_at_least_one_a, 3)

if __name__ == "__main__":
    n = int(input())
    letters = input().split()
    k = int(input())

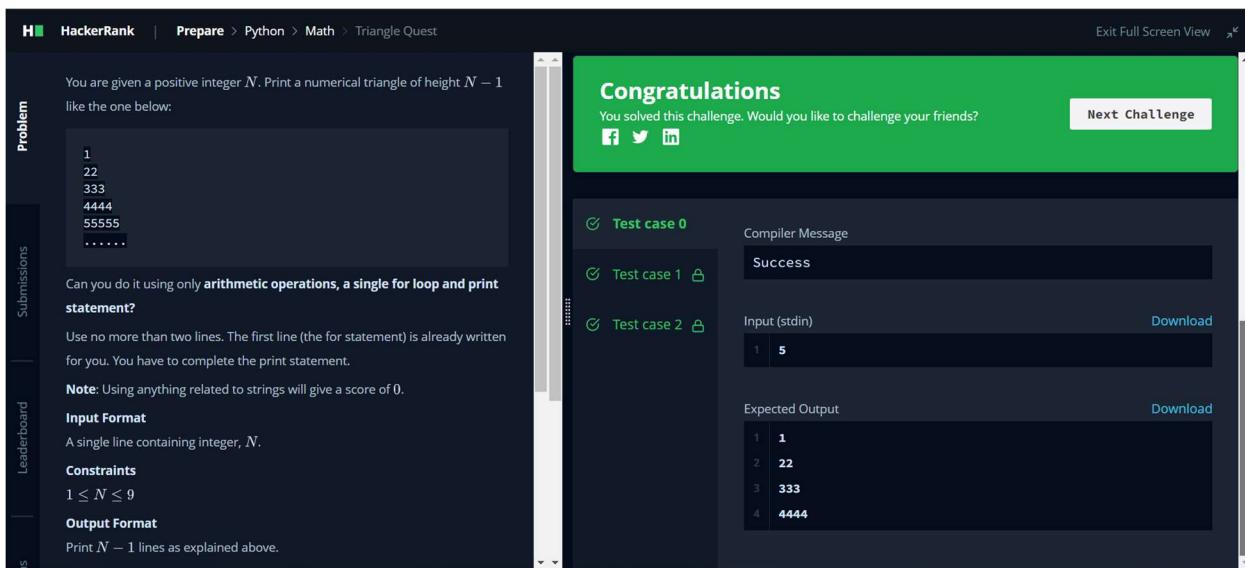
    result = calculate_probability(n, letters, k)
    print(result)
```

The screenshot shows the HackerRank challenge interface for the 'Iterables and Iterators' problem. The challenge description explains the use of the `itertools` module to calculate the probability of selecting at least one letter 'a' from a list of lowercase English letters. It provides documentation links and details about the input and output formats.

The challenge interface includes a sidebar with 'Submissions' and 'Leaderboard' tabs. The main area features a 'Congratulations' banner for solving the challenge. It shows the test cases (Test case 0 to Test case 6), a 'Compiler Message' indicating 'Success', and the 'Input (stdin)' and 'Expected Output' for each test case. The input for Test case 0 is 4, a a c d, 2, and the expected output is 0.8333333333333333.

13. Triangle Quest:

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



14. Classes: Dealing with Complex Numbers

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

    def __add__(self, other):
        return Complex(self.real + other.real, self.imaginary + other.imaginary)

    def __sub__(self, other):
        return Complex(self.real - other.real, self.imaginary - other.imaginary)

    def __mul__(self, other):
        real_part = self.real * other.real - self.imaginary * other.imaginary
        imag_part = self.real * other.imaginary + self.imaginary * other.real
        return Complex(real_part, imag_part)

    def __truediv__(self, other):
        conjugate = Complex(other.real, -other.imaginary)
        denominator = other * conjugate
        result = self * conjugate
        result.real /= denominator.real
        result.imaginary /= denominator.real
```

```

        return result

    def mod(self):
        return Complex((self.real**2 + self.imaginary**2)**0.5, 0)

    def __str__(self):
        return "% .2f%+.2fi" % (self.real, self.imaginary)

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')

```

The screenshot shows a challenge titled "Classes: Dealing with Complex Numbers" on the HackerRank platform. The left sidebar includes navigation links like "HackerRank", "Prepare", "Python", "Classes", and the current challenge. It also has sections for "Problem", "Submissions", and "Leaderboard". The main content area contains the challenge description, which states: "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." Below this are "Input Format" and "Output Format" instructions. The "Input Format" specifies a single line of input with two space-separated floating-point numbers. The "Output Format" specifies the output sequence for two complex numbers C and D . The challenge has been solved, as indicated by the "Congratulations" message at the top right, which says "You solved this challenge. Would you like to challenge your friends?". It also shows a "Compiler Message" of "Success", the "Input (stdin)" (two lines of integers), and the "Expected Output" (four lines of complex number strings). There are "Download" buttons for both input and output.

15. 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())

# Sort the array based on the k-th attribute
sorted_arr = sorted(arr, key=lambda x: x[k])

# Print the sorted array
for row in sorted_arr:
    print(*row)

```

The screenshot shows a challenge titled "Athlete Sort" on the HackerRank platform. The left sidebar includes navigation links for "HackerRank", "Prepare", "Python", "Built-Ins", and "Athlete Sort". A "Problem" tab is selected. On the right, a "Congratulations" message is displayed, stating "You solved this challenge. Would you like to challenge your friends?" with social sharing icons for Facebook, Twitter, and LinkedIn, and a "Next Challenge" button.

The challenge details section contains the following text:

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.

A table is shown with two columns of data:

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 .

The right panel shows the results of the submission. It displays "Test case 0" and "Test case 1" both showing "Success". The "Compiler Message" indicates "Success". The "Input (stdin)" field shows the input data, and the "Download" button is available.

16. ginortS

```

# Enter your code here. Read input from STDIN. Print output to STD
OUT
s = input()

# Sort the string based on the specified criteria
sorted_string = sorted(s, key=lambda x: (x.isdigit(), x.isdigit()
and int(x) % 2 == 0, x.isupper(), x))

```

```
# Print the sorted string
print(''.join(sorted_string))
```

You are given a string S .
 S contains alphanumeric characters only.

Sorting

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

Congratulations

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

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 Sorting1234

Expected Output Download
1 ginorts1324

17. Validating Email Addresses With a Filter

```
import re

def fun(s):
    # Define the pattern for a valid email address
    pattern = r'^[a-zA-Z0-9_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'
    return bool(re.match(pattern, s))

def filter_mail(emails):
    return list(filter(fun, emails))

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)
```

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

Next Challenge

Compiler Message

Success

Input (stdin)

1 3
2 lara@hackerrank.com
3 brian-23@hackerrank.com
4 britts_54@hackerrank.com

Expected Output

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

18. 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)
```

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
```

Input Format

First line contains n , the number of rational numbers.

Congratulations
You solved this challenge. Would you like to challenge your friends?
[Next Challenge](#)

Compiler Message
Success

Hidden Test Case
Unlock this testcase for 5 hackos.
[Unlock](#)

19. Regex Substitution

```
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)))
```

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()`.

Transformation of Strings

Code

```
import re

#Squaring numbers
def square(match):
    number = int(match.group(0))
    return str(number**2)

print(re.sub(r"\d+", square, "1 2 3 4 5 6 7 8 9"))
```

Output

```
1 4 9 16 25 36 49 64 81
```

Congratulations
You solved this challenge. Would you like to challenge your friends?
[Next Challenge](#)

Compiler Message
Success

Input (stdin)
Download

```
1 11
2 a = 1;
3 b = input();
4
5 if a + b > 0 && a - b < 0:
6     start()
7 elif a+b > 10 || a/b < 1:
8     stop()
9 print set(list(a)) | set(list(b))
```

20. Validating Credit Card Numbers

```
import re
```

```

def is_valid_credit_card(card_number):
    pattern = re.compile(
        r'^([4-6]\d{3}-?\d{4}-?\d{4}-?\d{4}|[4-6]\d{15})$'
    )
    if re.match(pattern, card_number):
        card_number = card_number.replace('-', '')
        if re.search(r'(\d)\1{3,}', card_number):
            return 'Invalid'
        else:
            return 'Valid'
    else:
        return 'Invalid'

def main():
    n = int(input())
    for _ in range(n):
        card_number = input().strip()
        result = is_valid_credit_card(card_number)
        print(result)

if __name__ == "__main__":
    main()

```

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.

Examples:

Valid Credit Card Numbers

```

4253625879615786
4424424424442444
5122-2368-7954-3214

```

Invalid Credit Card Numbers

```

4253625879615786      #17 digits in card number → Invalid

```

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

Compiler Message
Success

Input (stdin)

```

1 6
2 4123456789123456
3 5123-4567-8912-3456
4 61234-567-8912-3456
5 4123356789123456
6 5133-3367-8912-3456
7 5123 - 3567 - 8912 - 3456

```

21. Words Score

```

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))

```

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.

Input Format

The input is read by the provided locked code template. In the first line, there is a single integer n denoting the number of words. In the second line, there are n space-separated lowercase words.

Constraints

- $1 \leq n \leq 20$
- Each word has at most 20 letters and all letters are English lowercase letters

Congratulations
You solved this challenge. Would you like to challenge your friends?
[Next Challenge](#)

Test case 0	Compiler Message
Success	Success
Test case 1	
Test case 2	
Test case 3	
Test case 4	
Test case 5	
Test case 6	

Input (stdin) Expected Output
1 2 1 4
2 hacker book

Download Download

18 new notifications

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

        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())

```

The screenshot shows the HackerRank challenge interface for a Python debugging challenge. The left sidebar includes navigation links for HackerRank, Prepare, Python, Debugging, and Default Arguments. The main area displays the challenge details and the user's solution code.

Problem Description:

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

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

```

Congratulations

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

Test Results:

Test Case	Compiler Message
Test case 0	Success
Test case 1	Success
Test case 2	Success
Test case 3	Success
Test case 4	Success
Test case 5	Success
Test case 6	Success

Input (stdin)

```

1 3
2 odd 2
3 even 3
4 odd 5

```

Expected Output

```

1
2 3

```

Hard Difficulty Challenges:

1. Maximize It!

```
from itertools import product

from itertools import product

def find_max(k_lists, M):
    f = lambda x: x*x
    combinations = product(*k_lists)
    results = [(sum([f(a) for a in x]) % M) for x in combinations]
    # Return the maximum value
    return max(results)

K, M = tuple(map(int, input().split(" ")))
k_lists = [list(map(int, input().split(" ")))[:_] for _ in range(K)]
print(find_max(k_lists, M))
```

The screenshot shows the HackerRank challenge interface for the 'Maximize It!' problem. On the left, the problem statement describes the task: given a function $f(X) = X^2$ and K lists of integers, find the maximum value of the sum of squares of elements chosen from each list such that the result is less than or equal to M . It includes sample code and constraints. On the right, a 'Congratulations' message is displayed, stating 'You solved this challenge. Would you like to challenge your friends?' with social sharing icons. Below it, a 'Compiler Message' shows 'Success' for all test cases (Test case 0 to Test case 6). A 'Hidden Test Case' is mentioned with a 'Unlock' button.

2. 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)

```

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

Test case	Compiler Message
Test case 0	Success
Test case 1	
Test case 2	
Test case 3	
Test case 4	
Test case 5	
Test case 6	

3. Matrix Script

```

#!/bin/python3

import math
import os
import random
import re
import sys

m,n = tuple(map(int, input().split()))
a=[]
for i in range(m):
    a.append(list(input().rstrip('\n')))
s=""
for i in range(n):
    for j in range(m):
        s+=a[j][i]
p = re.compile(r'([a-zA-Z0-9][\$#\!@\ \%\&]{1,99}[a-zA-Z0-9])')
matches = list(map(lambda x: x[1:-1], re.findall(p, s)))
b=s[:]
for i in matches:

```

```
b=b.replace(i, " ", 1)  
  
print(b)
```

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

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.

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

[Next Challenge](#)

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)

1 7 3
2 Ts i
3 h o x
4 i #
5 s M
6 \$ a
7 # t %
8 i r !

Download