NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

School of Mechanical and Manufacturing Engineering

Artificial Intelligence (CSE-860)

ASSIGNMENT# 03

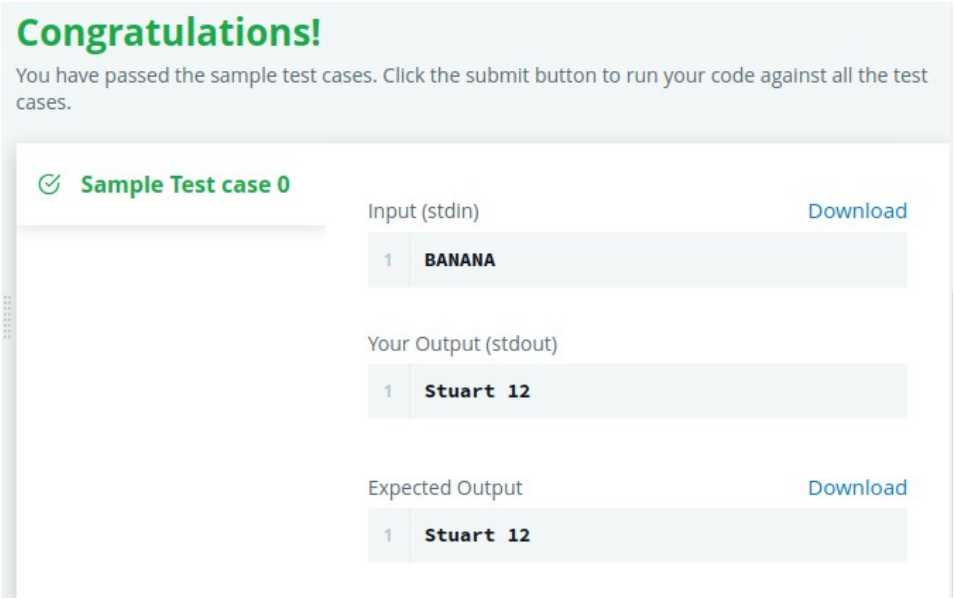
SUBMITTED TO:	**Dr Yasar Ayaz**
SUBMITTED BY:	**Muhammad Saqib (126)**
Roll No.	**482486**
PROGRAMME	**PhD-RIME**
Date of Submission:	**01 Jan, 2024**

Difficulty Level: Medium

1. The Minion Game

```
def get_all_substrings(string):
    """Returns a list of all possible substrings of the given string."""
    substrings = []
    for i in range(len(string)):
        for j in range(i, len(string)):
            substrings.append(string[i:j + 1])
    return substrings

def minion_game(string):
    # your code goes here
    a2z = 'abcdefghijklmnopqrstuvwxyz'.upper()
    vow = 'AEIOU' #Kevin
    con = 'BCDFGHJKLMNPQRSTVWXYZ' #Stuart
    kpoints = 0
    spoints = 0
    substrings = get_all_substrings(string)
    # print(substrings)
    for subst in substrings:
        for char in vow:
            if subst.startswith(char):
                kpoints += 1
        for char in con:
            if subst.startswith(char):
                spoints += 1
    if kpoints == spoints:
        print('Draw')
    elif kpoints > spoints:
        print('Kevin', kpoints)
    else:
        print('Stuart', spoints)

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

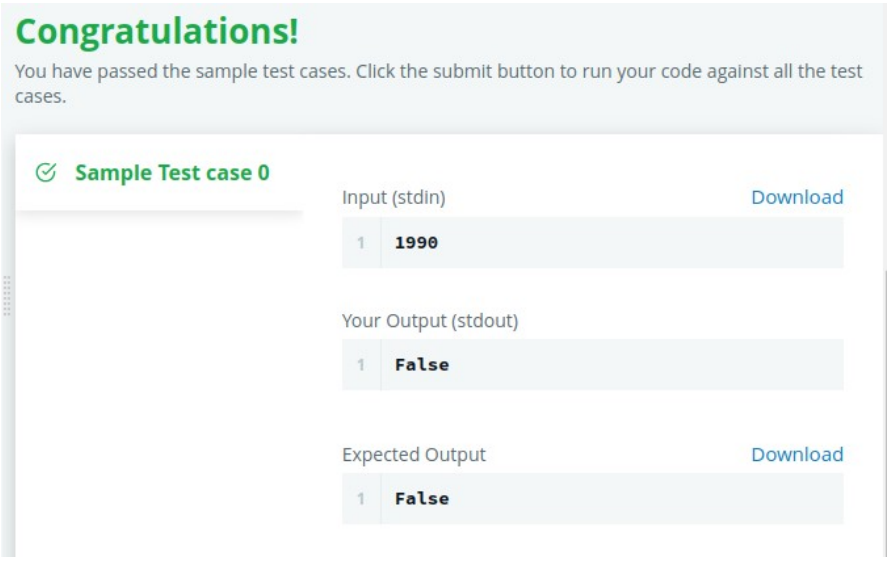
Output:



2. Write a function

```
def is_leap(year):
    leap = False
    # Write your logic here
    if year % 4 == 0 and year % 400 == 0:
        return True
    else:
        return False

    return leap

year = int(input())
print(is_leap(year))
```

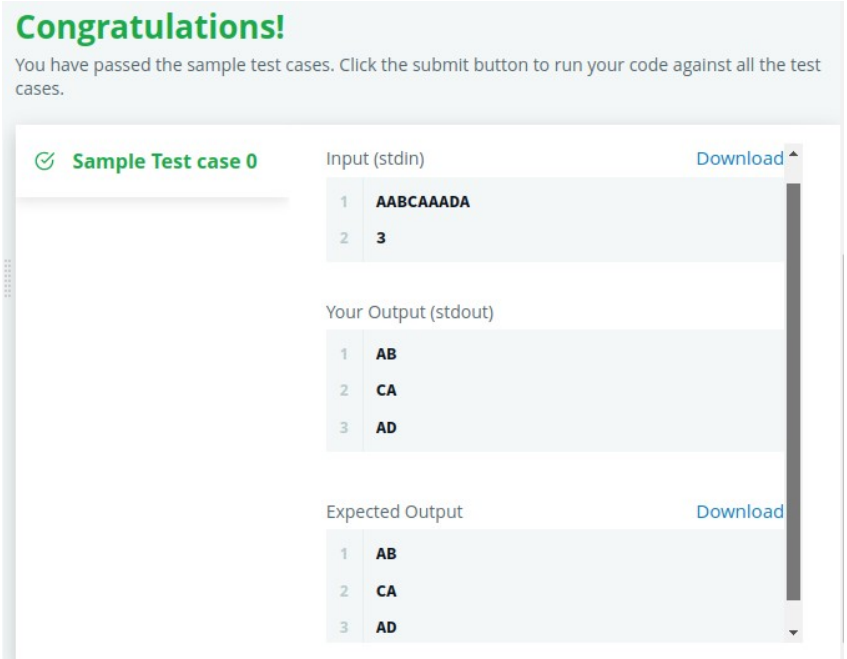
Output:



3. Merge the Tools!

```
def merge_the_tools(string, k):
    # your code goes here
    x, x2, result, current_group, y = [], [], [], [], []
    n = len(string)

    for i in range(0, n):
        x.append(string[i])

    for i, item in enumerate(x, start=1):
        x2.append(item)
        if i % (n//(n//k)) == 0 and i < len(x):
            x2.append('-')


    for item in x2:
        if item == "-":
            result.append(current_group)
            current_group= []

        else:
            current_group.append(item)

    result.append(current_group)

    for sublist in result:
        unique_elements = []
        for item in sublist:
            if item not in unique_elements:
                unique_elements.append(item)
        y.append(unique_elements)

    for i in y:
        for j in i:
            print(j, end="")
        print("")

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

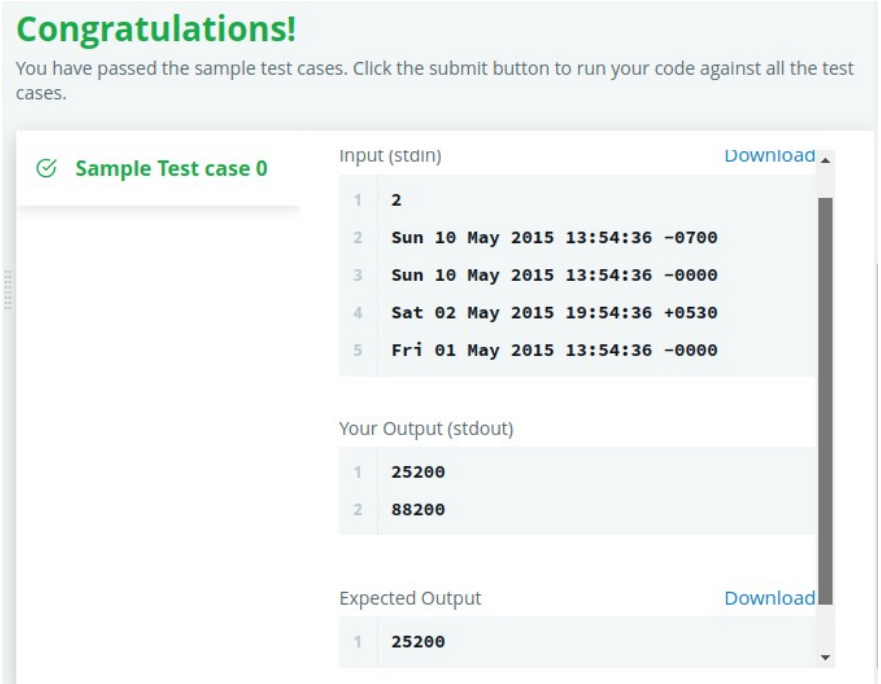
Output:


Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.
Sample Test case 0
Input (stdin)
Download
1 AABCAAADA
2 3
Your Output (stdout)
1 AB
2 CA
3 AD
Expected Output
Download
1 AB
2 CA
3 AD

4. Time Delta

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

# Complete the time_delta function below.
def time_delta(t1, t2):
    t1_date = datetime.strptime(t1, '%a %d %b %Y %H:%M:%S %z')
    t2_date = datetime.strptime(t2, '%a %d %b %Y %H:%M:%S %z')
    if t1_date > t2_date:
        time_diff = t1_date - t2_date
    else:
        time_diff = t2_date - t1_date
    if time_diff.days == 0:
        time_diff_seconds = time_diff.seconds
    else:
        time_diff_seconds = time_diff.days * 24 *3600 + time_diff.seconds
    return str(time_diff_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()
```

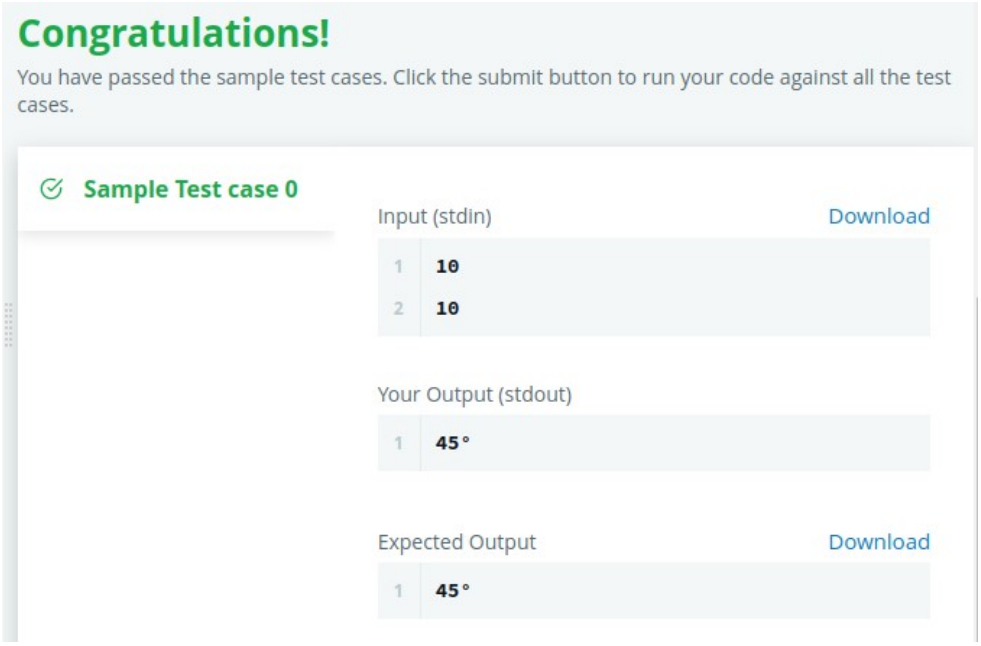
Output:


Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.
Sample Test case 0
Input (stdin)
Download
2
Sun 10 May 2015 13:54:36 -0700
Sun 10 May 2015 13:54:36 -0000
Sat 02 May 2015 19:54:36 +0530
Fri 01 May 2015 13:54:36 -0000
Your Output (stdout)
25200
88200
Expected Output
Download
25200

5. Find Angle MBC

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

import math
print(f'{round(math.degrees(math.atan(int(input())/int(input()))))}\u00b0')
```

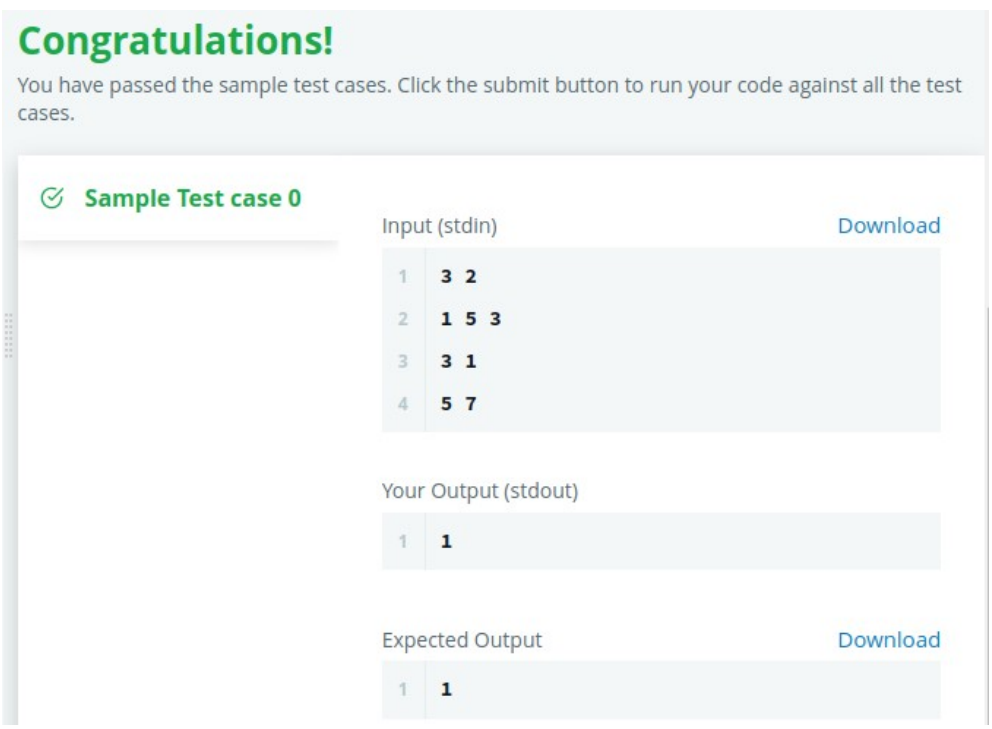
Output:



6. No Idea!

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

happiness = 0

#Im dealing with them as string not like INT
n,m = map(int, input().split()) #3 2
happiness_nos = input().split()
A = set(input().split()) #3 1
B = set(input().split()) #5 3

if n != len(happiness_nos) and m != len(A) and m != len(B) :
    print('Size Mismatch')

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

print(happiness)
```

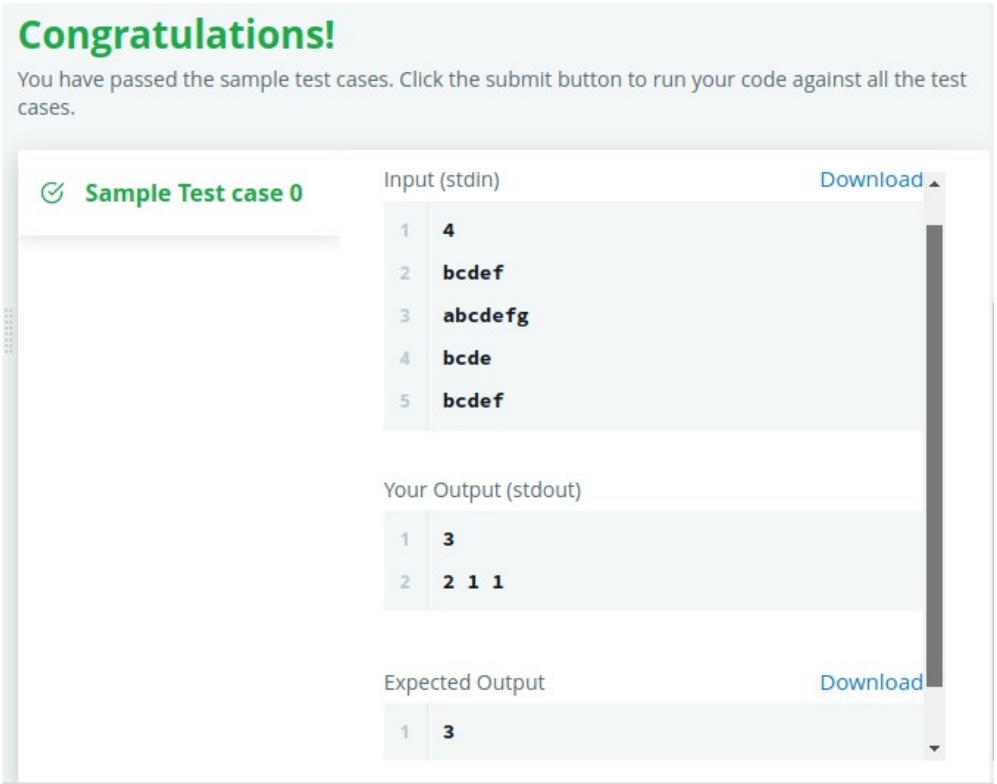
Output:



7. Word Order

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

x = int(input())
word_dict = {}
for i in range(x):
    word = input()
    if word in word_dict.keys():
        word_dict[word] += 1
    else:
        word_dict[word] = 1
print(len(word_dict))
for v in word_dict.values():
    values_list = []
    values_list.append(str(v))
    print(' '.join(values_list), end=' ')
```

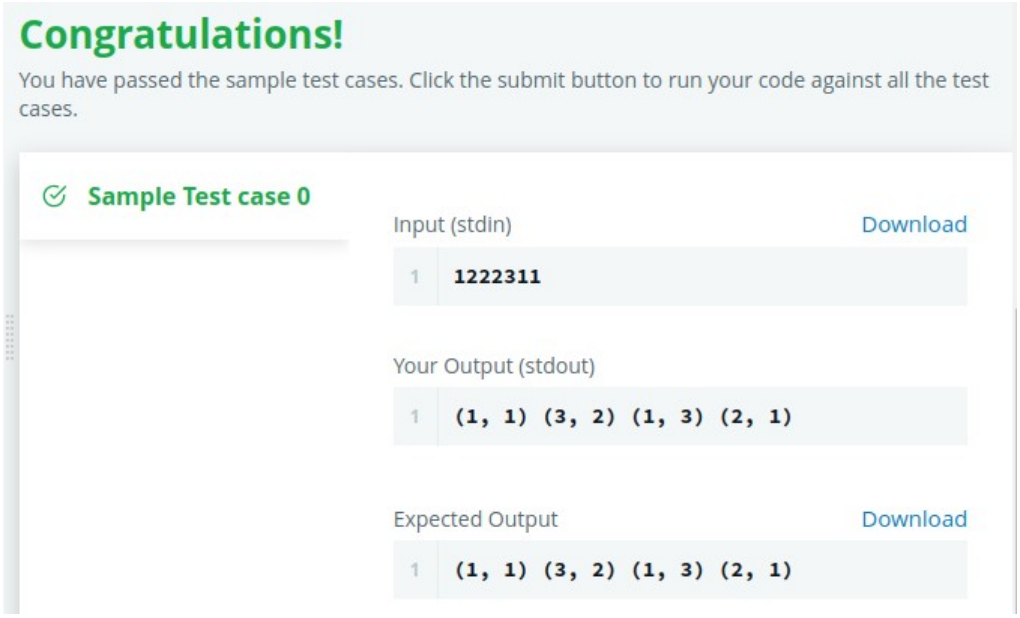
Output:



8. Compress the String!

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

from itertools import groupby
string = str(input())
lis = [list(g)for k,g in groupby(string)]
result = []
for n in range(len(lis)):
    result.append([lis[n].count(lis[n][0]),int(lis[n][0])])

final = []
for x in range(len(result)):
    final.append(str("({occurance}, {key})".format(occurance=result[x][0],key=result[x][1])))
print(*final)
```

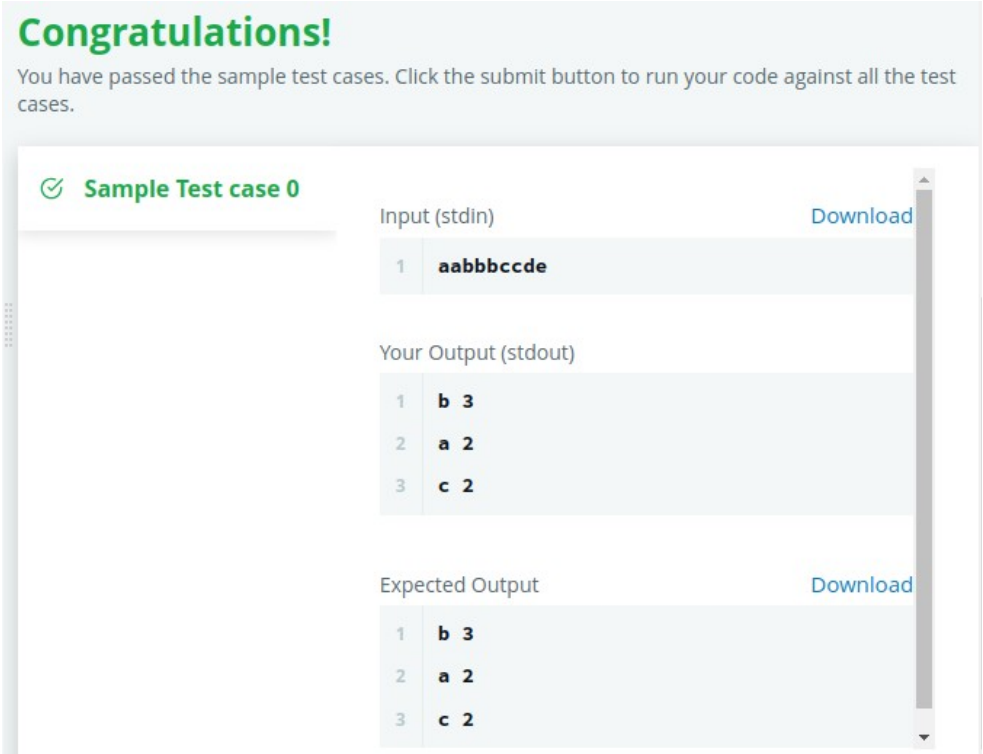
Output:



9. Company Logo

```
#!/bin/python3

import math
import os
import random
import re
import sys

from collections import Counter

if __name__ == '__main__':
    s = input()
    char_count = (Counter(s))
    sorted_chars = sorted(char_count.items(), key=lambda x: (-x[1], x[0]))
    for char, count in sorted_chars[:3]:
        print(f"{char} {count}")
```

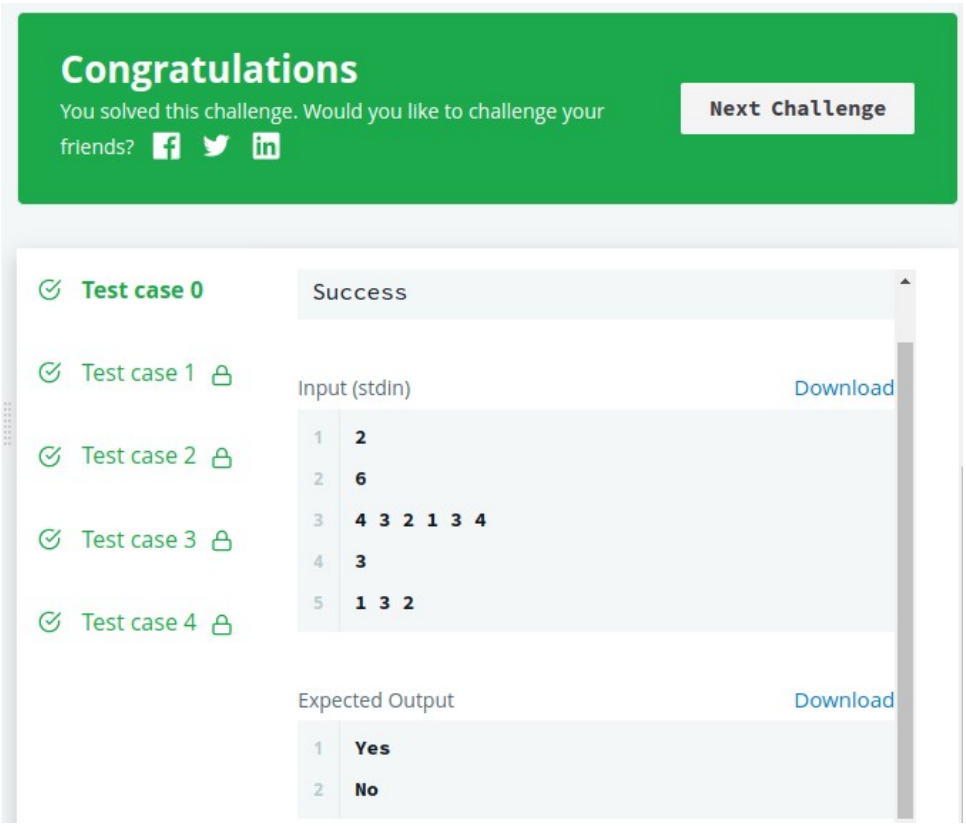
Output:



10. Piling Up!

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

from collections import deque
tC = int(input())
for _ in range(tC):
    i, d, li = int(input()), deque(map(int, input().split())), []
    for _ in range(i):
        if d[-1] >= d[0]:
            li.append(d.pop())
        else:
            li.append(d.popleft())
    print("Yes" if li == sorted(li, reverse=True) else "No")
```

Output:



11. Triangle Quest 2

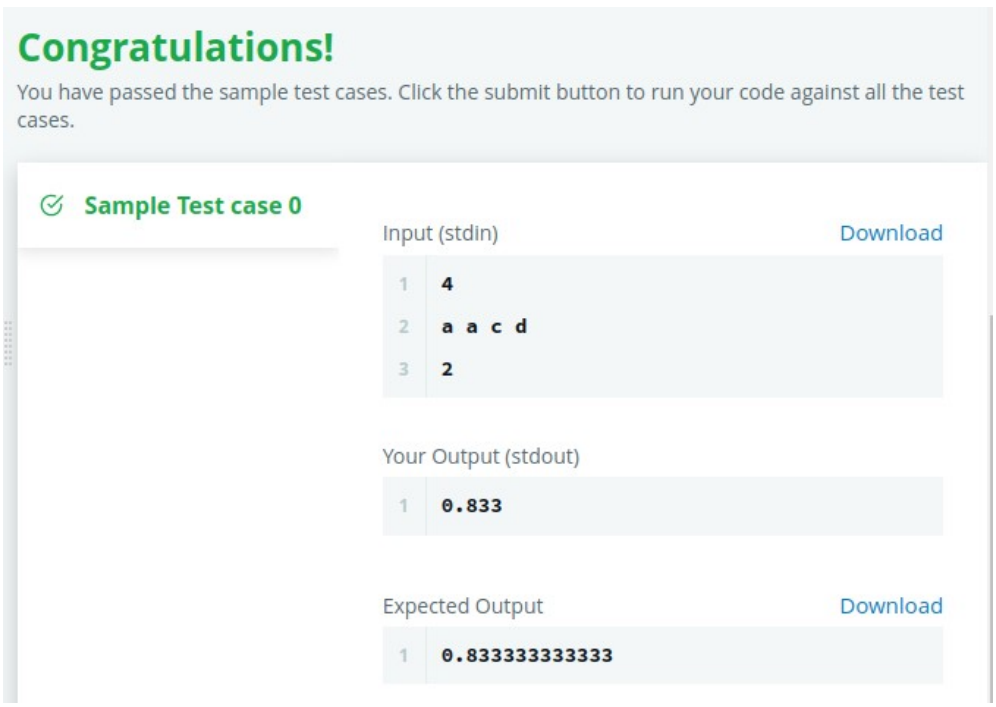
```
# Enter your code here. Read input from STDIN. Print output to STDOUT

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

Output:

12. Iterables and Iterators

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

from itertools import combinations

n = int(input())
lowercase_letters = list(input().split())
k = int(input())

# Create combinations using itertools
pos_comb = combinations(lowercase_letters, k)

# Count favorable outcomes
favorable_outcomes = sum('a' in x for x in pos_comb)

# Calculate probability
probability = favorable_outcomes / len(list(combinations(lowercase_letters, k)))

# Print the result with 3 decimal places
print(round(probability, 3))
```

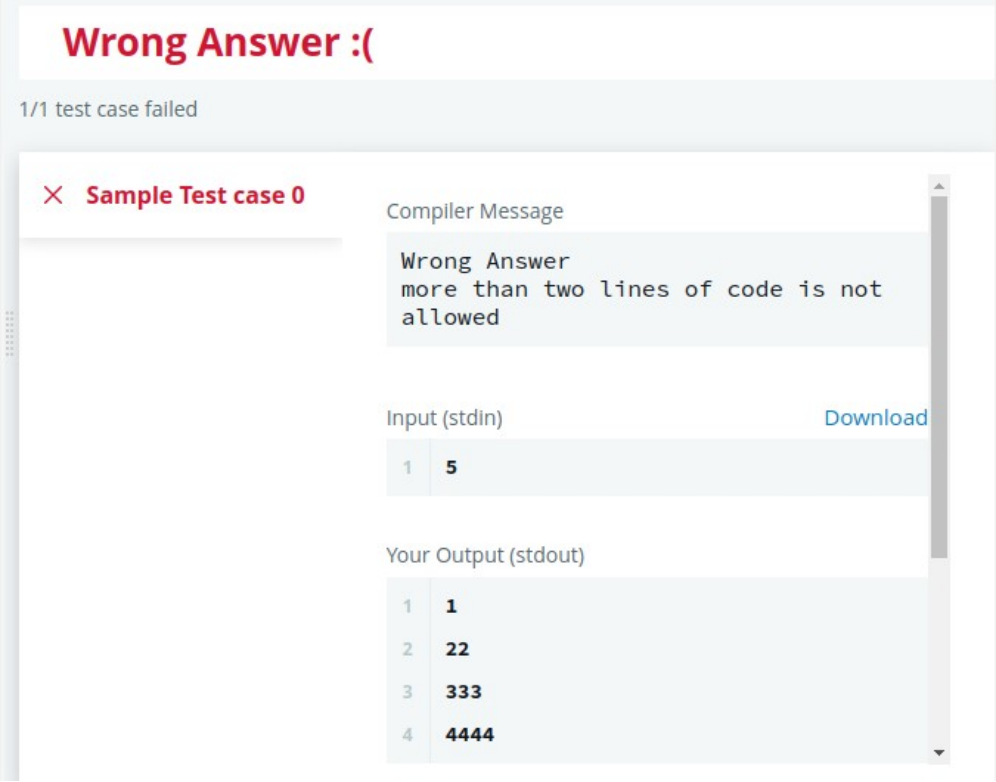
Output:



13. Triangle Quest

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

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

Output:



14. Classes: Dealing with Complex Numbers

```
import math

class Complex(object):
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary
        self.num = complex(real, imaginary)
    def __add__(self, no):
        n = self.num + complex(no.real, no.imaginary)
        return Complex(n.real, n.imag)
    def __sub__(self, no):
        n = self.num - complex(no.real, no.imaginary)
        return Complex(n.real, n.imag)
    def __mul__(self, no):
        n = self.num * complex(no.real, no.imaginary)
        return Complex(n.real, n.imag)
    def __truediv__(self, no):
        n = self.num / complex(no.real, no.imaginary)
        return Complex(n.real, n.imag)
    def mod(self):
        n = math.sqrt(self.real**2 + self.imaginary**2)
        return Complex(n.real, 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')
```

Output:

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

```
2 1
5 6
```

Your Output (stdout)

```
7.00+7.00i
-3.00-5.00i
4.00+17.00i
0.26-0.11i
2.24+0.00i
7.81+0.00i
```

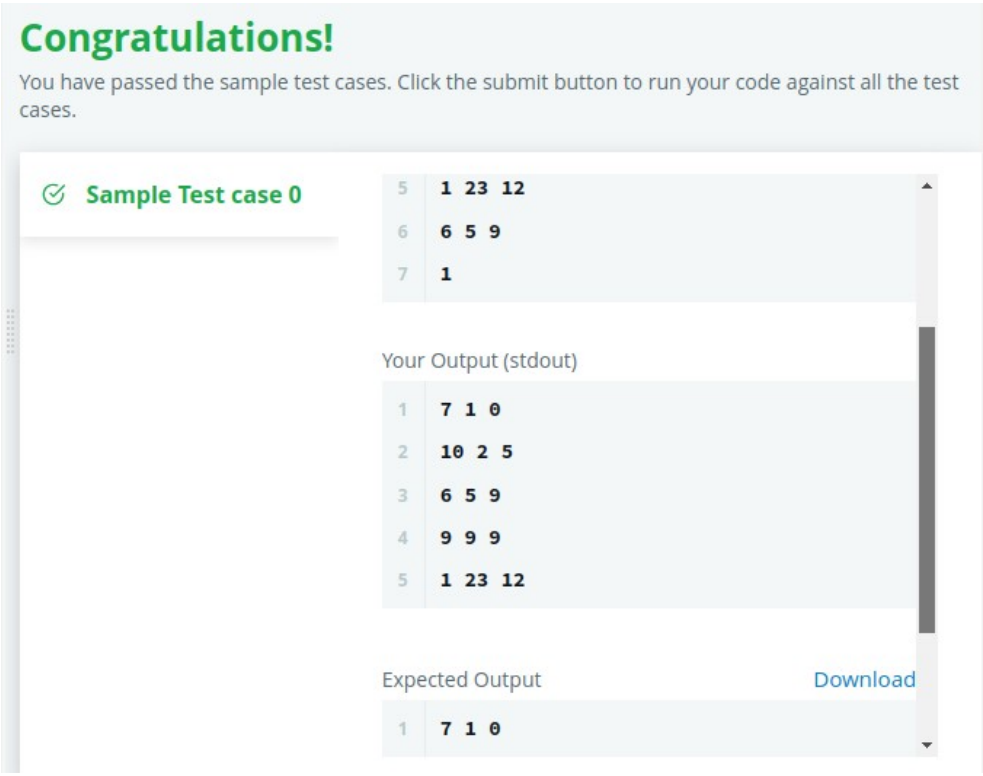
Expected Output Download

```
7.00+7.00i
```

15. Athlete Sort

```
#!/bin/python3

import math
import os
import random
import re
import sys


if __name__ == '__main__':
    n, m = map(int, input().split())

    arr = []

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

    k = int(input())
    for i in sorted(arr, key=lambda x: x[k]):
        print(*i)
```

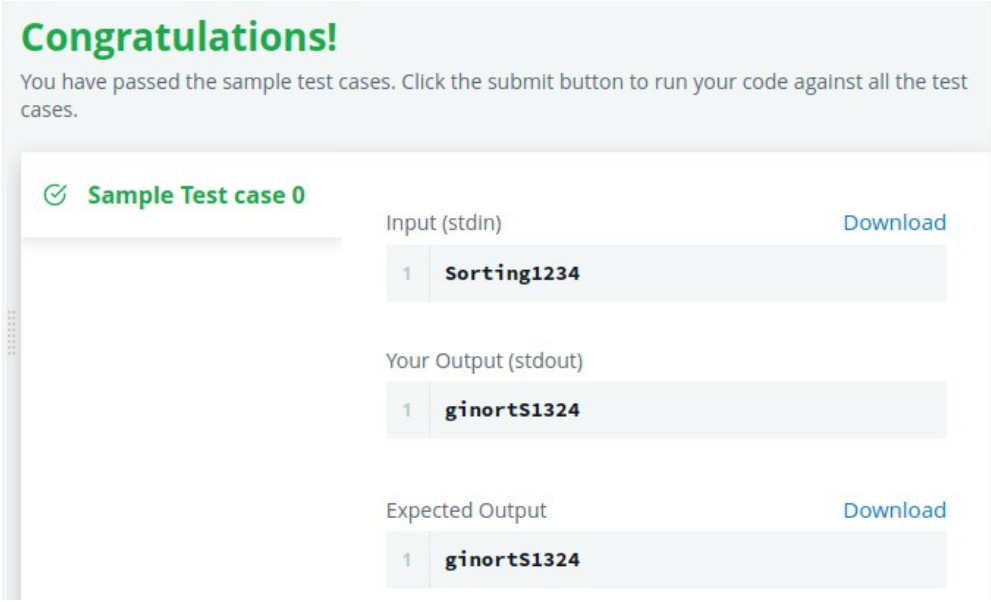
Output:



16. ginortS

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

from string import ascii_lowercase, ascii_uppercase, digits
from collections import OrderedDict

org_order = digits + ascii_uppercase + ascii_lowercase
new_order = ascii_lowercase + ascii_uppercase + '1357902468'
char2key = OrderedDict(zip(new_order, org_order))

def key_func(ch:str) -> str:
    if not ch.isalnum():
        return ch
    return char2key[ch]

print(''.join(sorted(input(), key=key_func)))
```

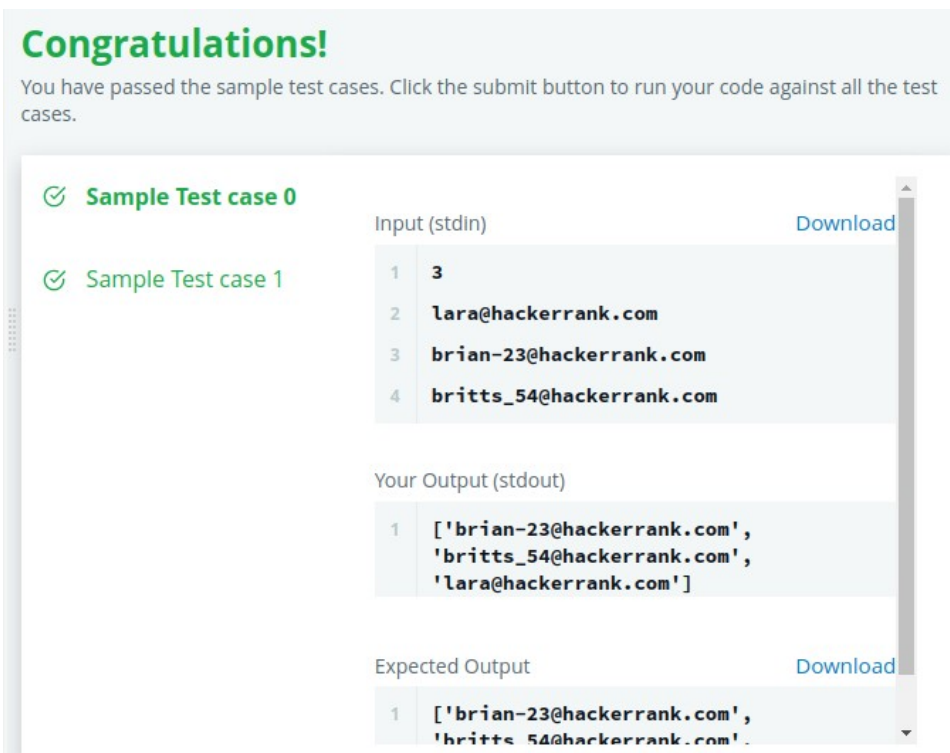
Output:



17. Validating Email Address With a Filter

```
def fun(s):
    # return True if s is a valid email, else return False
    if s.count('@') != 1 or s.count('.') != 1: return False
    un, web = s.split('@')
    wn, ext = web.split('.')
    return un.replace('-', '').replace('_', '').isalnum() and wn.isalnum() and ext.isalpha() and len(ext) <= 3

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)
```

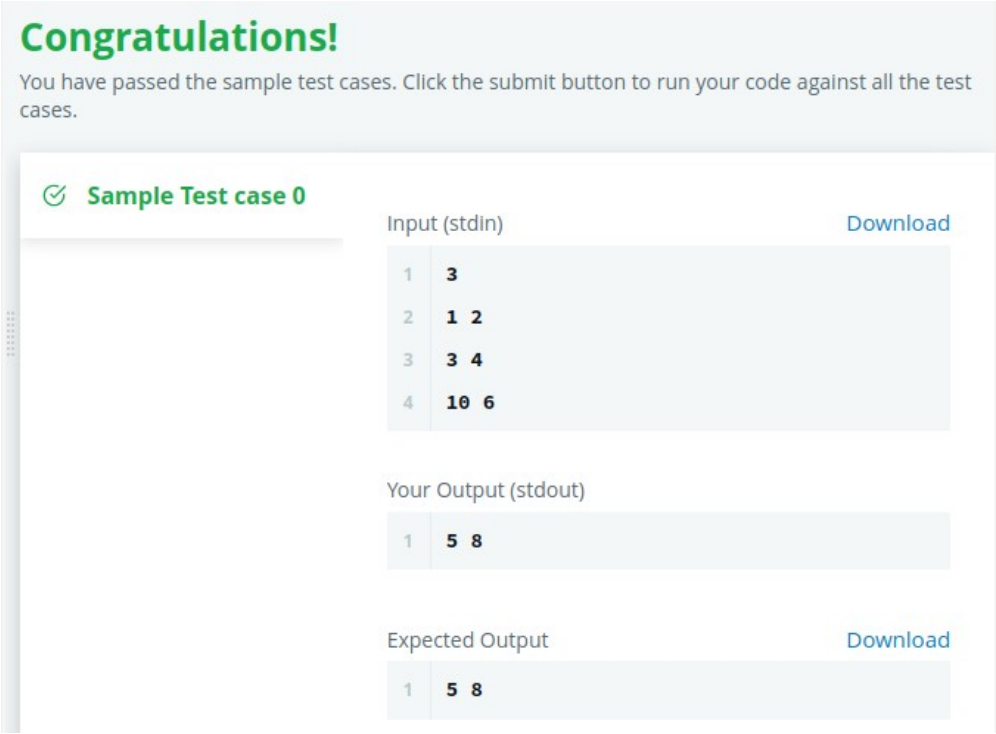
Output:



18. Reduce Function

```
from fractions import Fraction
from functools import reduce

def product(fracs):
    # t = # complete this line with a reduce statement
    t = reduce(lambda x,y : x * y, fracs, 1);
    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)
```

Output:



19. Regex Substitution

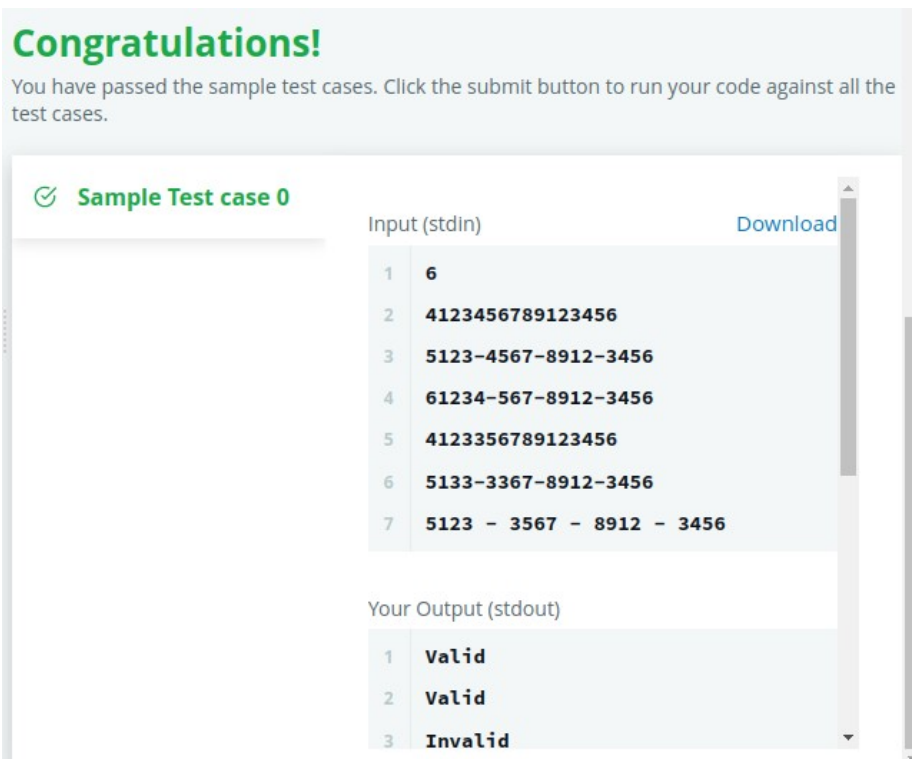
```
# Enter your code here. Read input from STDIN. Print output to STDOUT

import re

lines = [input() for _ in range(int(input()))]

print(re.sub(r'(?<= )\|\|(?= )', 'or', re.sub(r'(?<= )&&(?= )', 'and', '\n'.join(lines))))
```

Output:

20. Validating Credit Card Numbers

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

import re

# I know this isn't pretty, but I finally got something to work so i'll take it
cards = int(input())
card_list = [input() for x in range(0, cards)]

for card in card_list:
    verify_card = '^[4-6][0-9]{15}|^[4-6][0-9]{3}-[0-9]{4}-[0-9]{4}-[0-9]{4}$'
    if re.match(verify_card, card):
        card = card.replace("-", "")
        bad = False
        for num in range(0, 13):
            if int(card[num]) == int(card[num + 1]) == int(card[num + 2]) == int(card[num + 3]):
                bad = True
        if bad:
            print("Invalid")
        else:
            print("Valid")
    else:
        print("Invalid")
```

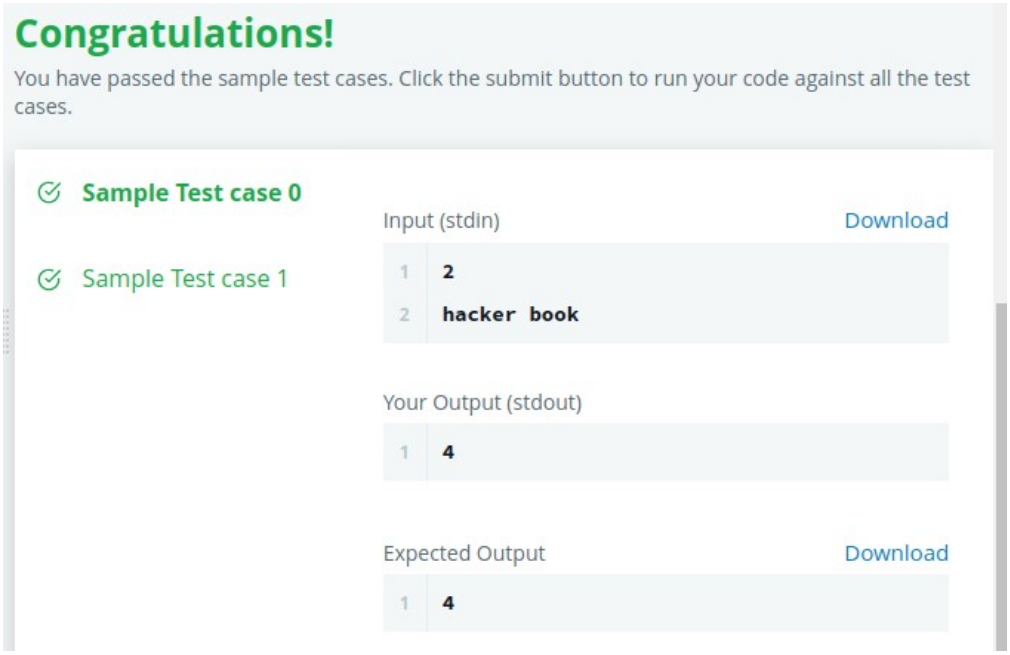
Output:



21. Word Score

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

alphabet = ['a','e','i','o','u','y']

def score_words(arry):
    count = 0
    score = []

    for word in arry:
        for chac in word:
            if chac in alphabet:
                count += 1
            else:
                pass
        if (count%2) == 0:
            score.append(2)
        else:
            score.append(1)
        count = 0

    return sum(score)
n = int(input())
user = input()
user = user.lower()
word_arry = user.split()

print(score_words(word_arry))
```

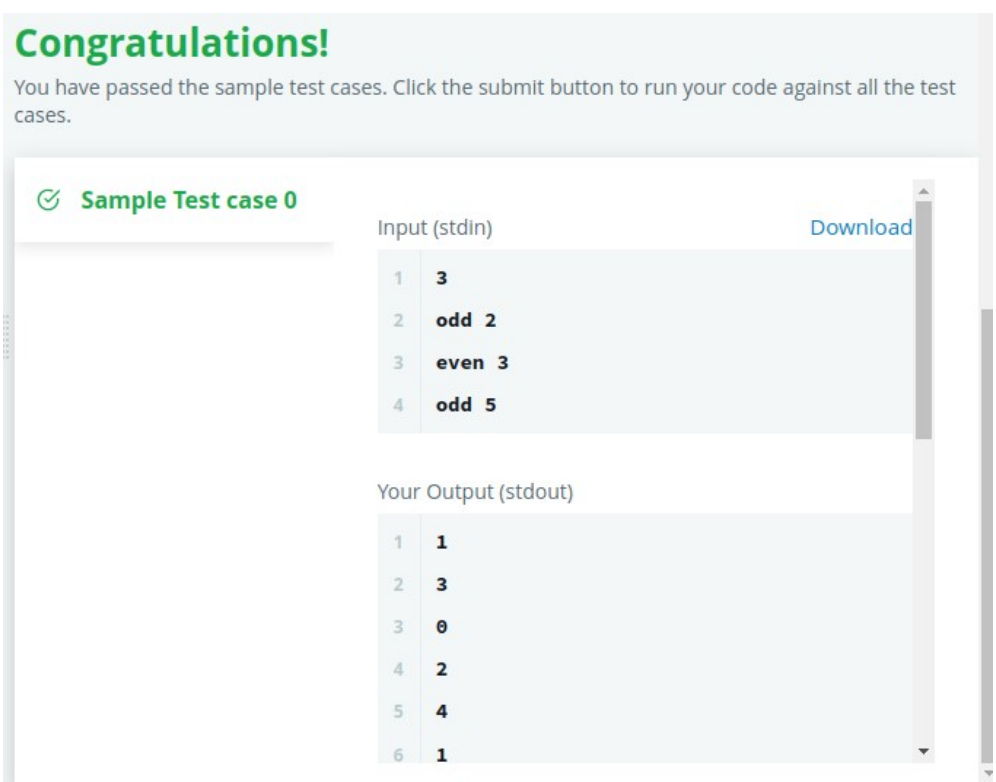
Output:



22. Default Arguments

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

def oddStream(n, base=1):
    if n == 0:
        return
    print(base)
    return oddStream(n-1, base=base+2)

def evenStream(n, base=0):
    if n == 0:
        return
    print(base)
    return oddStream(n-1, base=base+2)

def print_from_stream(n, stream):
    if stream == 'even':
        return evenStream(n)
    elif stream == 'odd':
        return oddStream(n)

number_of_lines = int(input())

for _ in range(number_of_lines):
    line = input().split()
    stream, n = line[0], int(line[1])
    print_from_stream(n, stream)
```

Outputs:

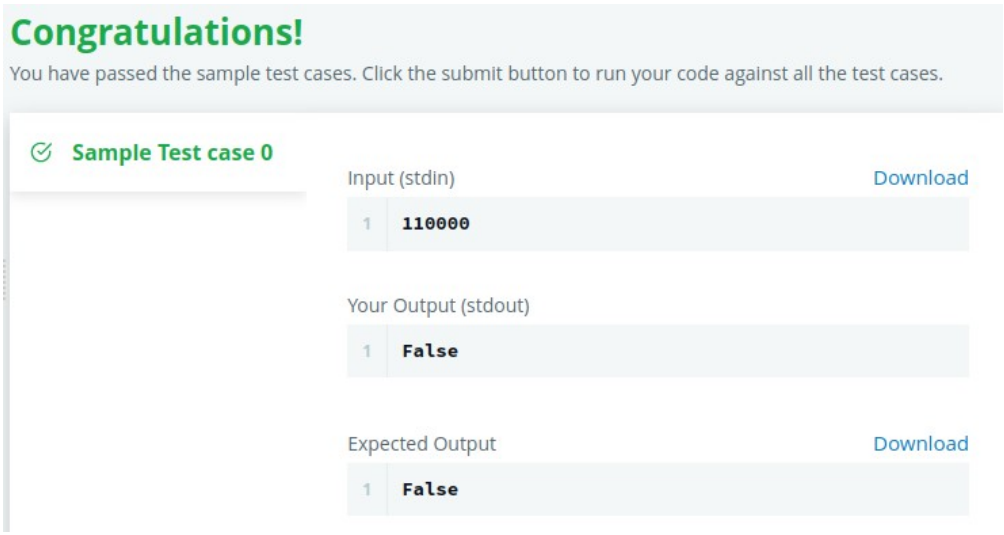


Difficulty Level: Hard

1. Validating Postal Codes

```
regex_integer_in_range = r"^[1-9][0-9]{5}$" # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"([0-9])(?=\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)
```

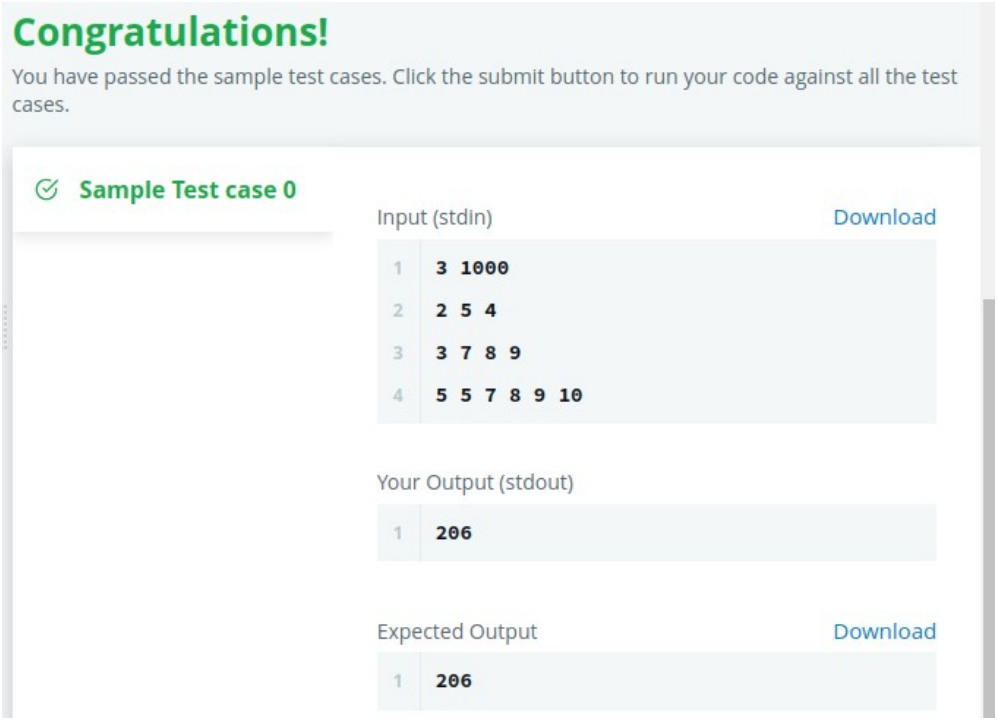
Output:



2. Maximize it

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

from itertools import product

square = lambda x: x**2
module = lambda x: sum(x) % M

if __name__ == "__main__":
    K, M = list(map(int, input().split()))
    arr = []
    for _ in range(K):
        arr.append(set(list(square(x) for x in map(int, input().split()))[1:]))

    print(max([module(x) for x in product(*arr)]))
```

Output:



3. Matrix Script

```python
import re


def decode(code: str) -> str:
    """If there are symbols or spaces between two alphanumeric characters
    of the decoded script, then the function replaces them with a single space "
    for better readability.
    The function doesn't use 'if' conditions for decoding.
    Alphanumeric characters consist of: [A-Z, a-z, and 0-9].
    >>> decode("This$#is% Matrix# %!")
    This is Matrix# %!
    >>> decode("This%%isMatrix#scrpt&%!&")
    This isMatrix scrpt&%!&
    """
    # Define a regular expression pattern to match non-alphanumeric characters
    # between alphanumeric characters
    pattern = r"(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])"

    # Use re.sub() to replace matched patterns with a single space
    decoded = re.sub(pattern, " ", code)

    # Return the decoded string
    return decoded

# Read the number of rows and columns from input
rows, columns = map(int, input().split())

# Read the matrix of characters row by row
matrix_dec, str_decoded = [input() for _ in range(rows)], ""

# Iterate through columns and rows to create a string by combining characters from the matrix
for col in range(columns):
    for row in range(rows):
        str_decoded += matrix_dec[row][col]

# Print the result of decoding the created string
print(decode(str_decoded))
```

Output:

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

```
h%x
i #
sM
$a
#t%
ir!
```

Your Output (stdout)

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

Expected Output Download

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