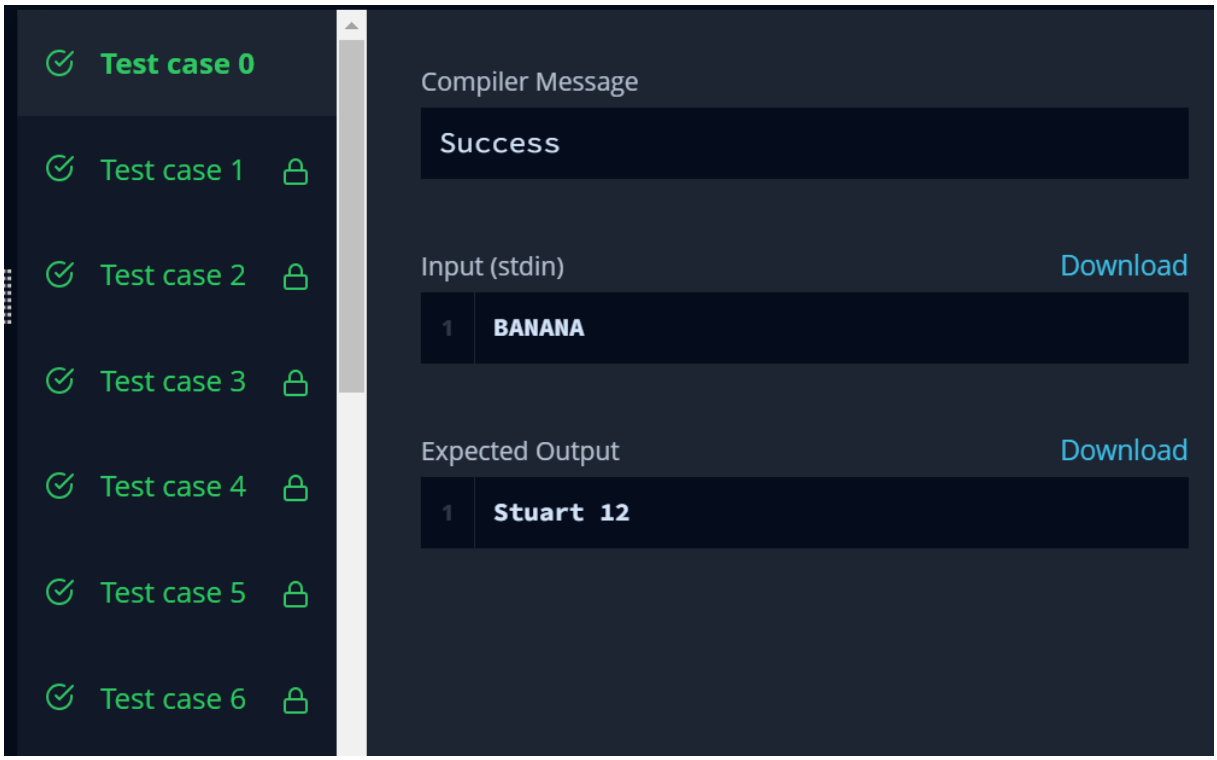
1. Medium Challenges

1.1. The Minion Game

```python
def minion_game(string):
    vowels = "AEIOU"
    kevin_score = 0
    stuart_score = 0

    for i in range(len(string)):
        if string[i] in vowels:
            kevin_score += len(string) - i
        else:
            stuart_score += len(string) - i

    if kevin_score > stuart_score:
        print("Kevin", kevin_score)
    elif stuart_score > kevin_score:
        print("Stuart", stuart_score)
    else:
        print("Draw")

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



1.2. Leap Year

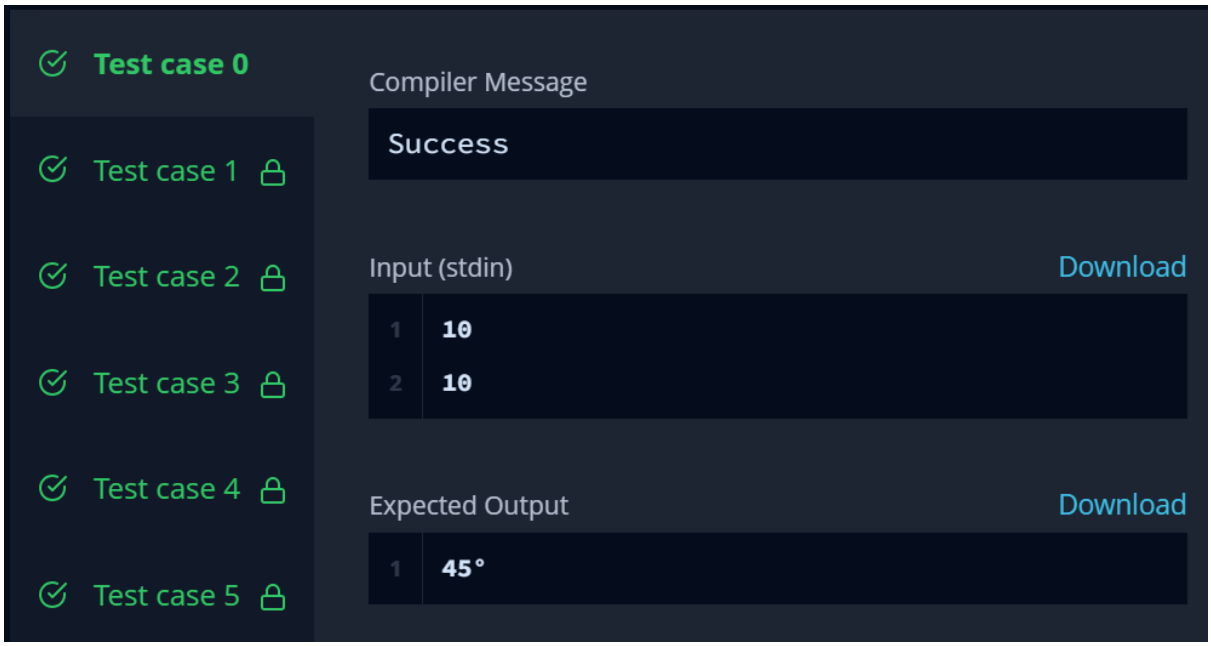
```
def is_leap(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False


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

1.3. Time Difference in Seconds

```
from datetime import datetime, timedelta

def time_difference(t1, t2):
    format_str = "%a %d %b %Y %H:%M:%S %z"
    time1 = datetime.strptime(t1, format_str)
    time2 = datetime.strptime(t2, format_str)
    return int(abs((time1 - time2).total_seconds()))
def main():
    n = int(input())
    for _ in range(n):
        t1 = input().strip()
        t2 = input().strip()
        result = time_difference(t1, t2)
        print(result)
if __name__ == "__main__":
    main()
```

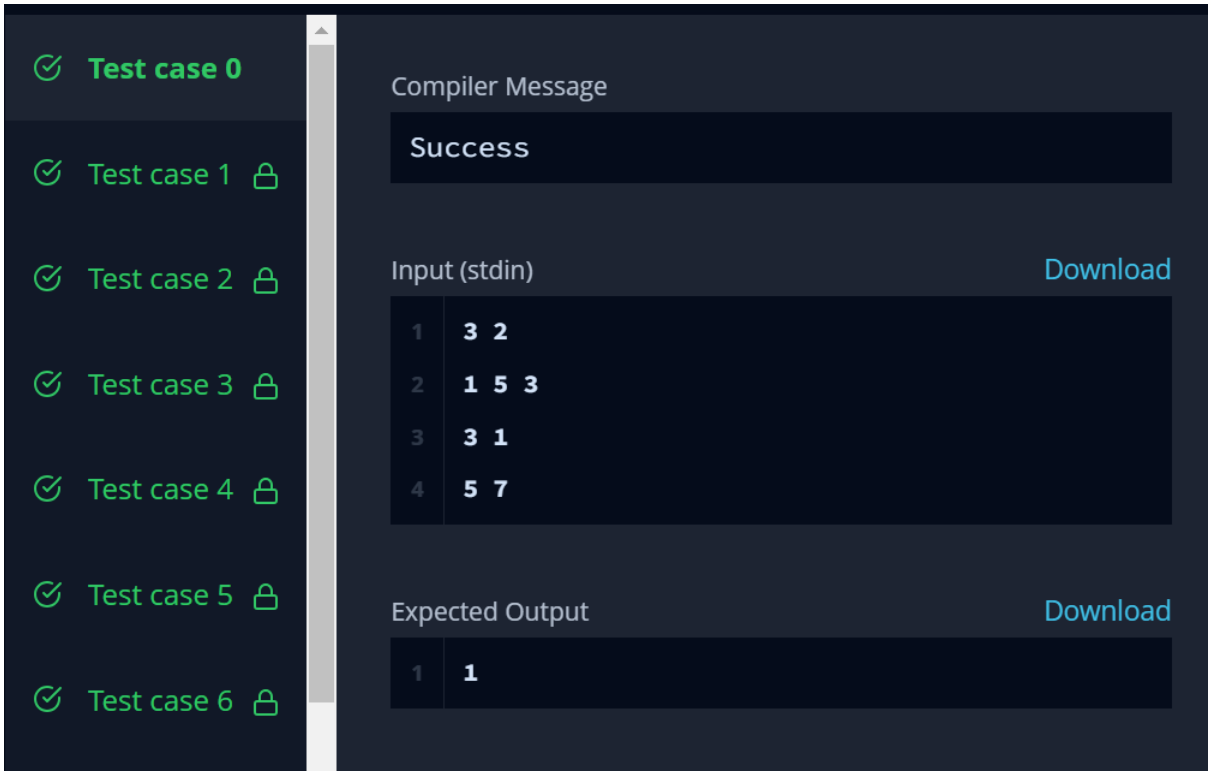
1.4. Find Angle

```
#!/usr/bin/env python3
from math import atan
from math import degrees
if __name__ == "__main__":
    ab = int(input().strip())
    bc = int(input().strip())
    print(u'{}\N{DEGREE SIGN}'.format(int(round(degrees(atan(ab/b
c))))))
```



1.5. No Idea!

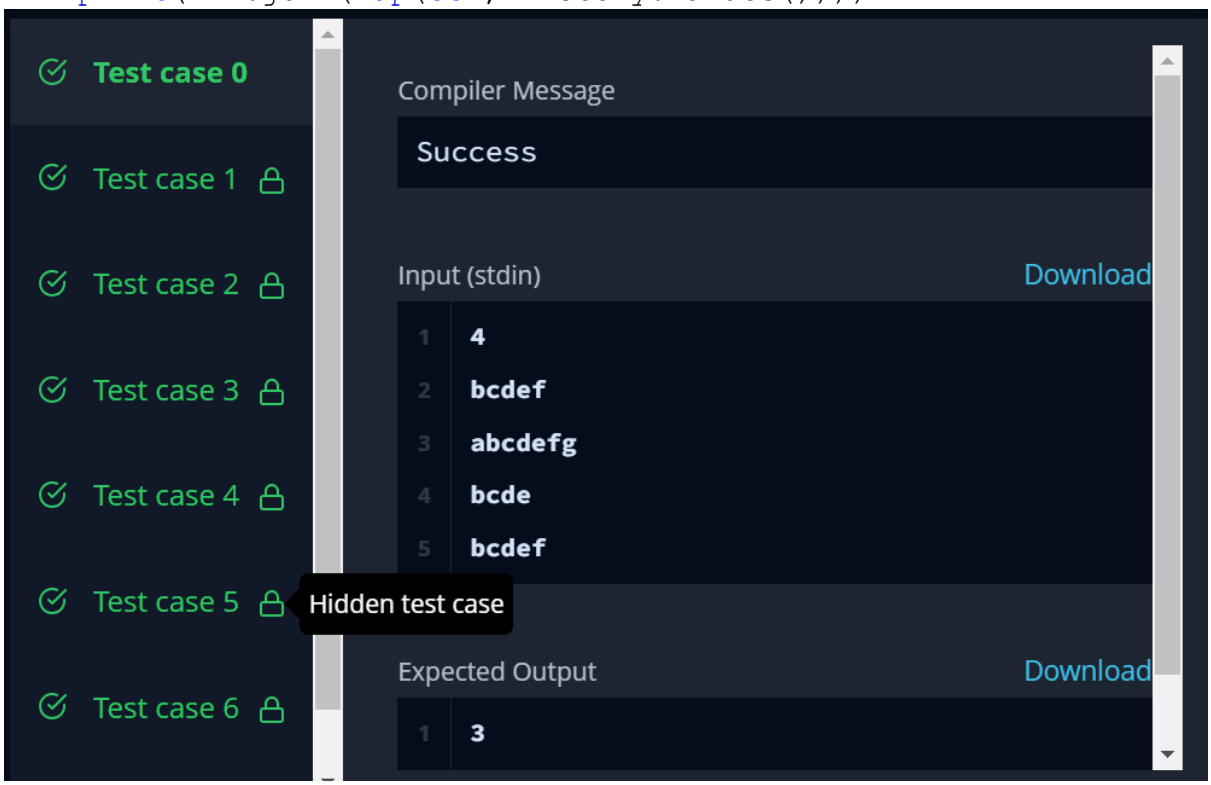
```python
#!/usr/bin/env python3
if __name__ == "__main__":
    happiness = 0
    n, m = map(int, input().strip().split(' '))
    arr = list(map(int, input().strip().split(' ')))

    good = set(map(int, input().strip().split(' ')))
    bad = set(map(int, input().strip().split(' ')))

    for el in arr:
        if el in good:
            happiness += 1
        elif el in bad:
            happiness -= 1

    print(happiness)
```



1.6. Word Order

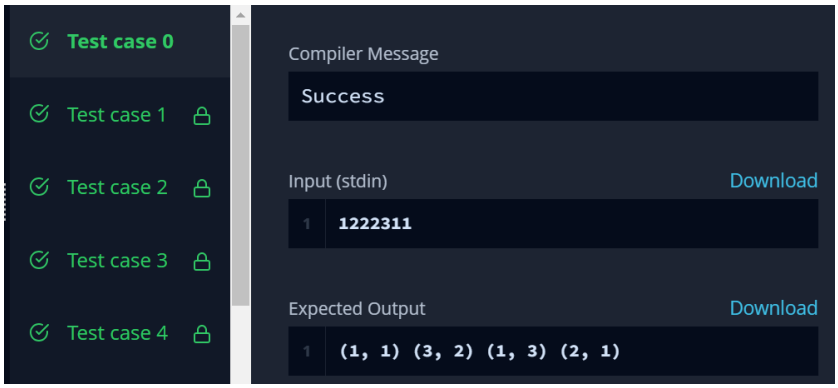
```python
#!/usr/bin/env python3
from collections import OrderedDict
if __name__ == "__main__":
    num = int(input().strip())
    history = OrderedDict()
```

```
    for _ in range(num):
        word = str(input().strip().split())
        if word not in history.keys():
            history[word] = 1
        else:
            history[word] += 1
    print(len(history.keys()))
    print(" ".join(map(str, history.values())))
```



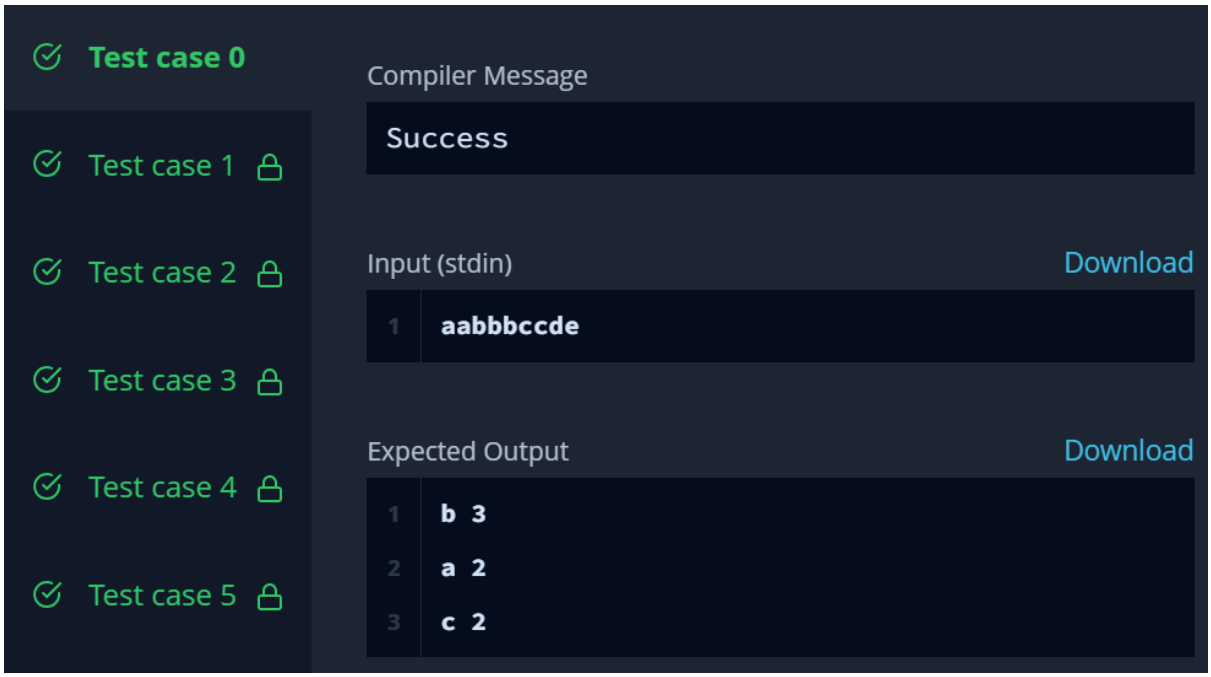
1.7. Compress the String!

```
#!/usr/bin/env python3
from itertools import groupby
if __name__ == "__main__":
    for el, el_list in groupby(input()):
        print((len(list(el_list)), int(el)), end=' ')
```



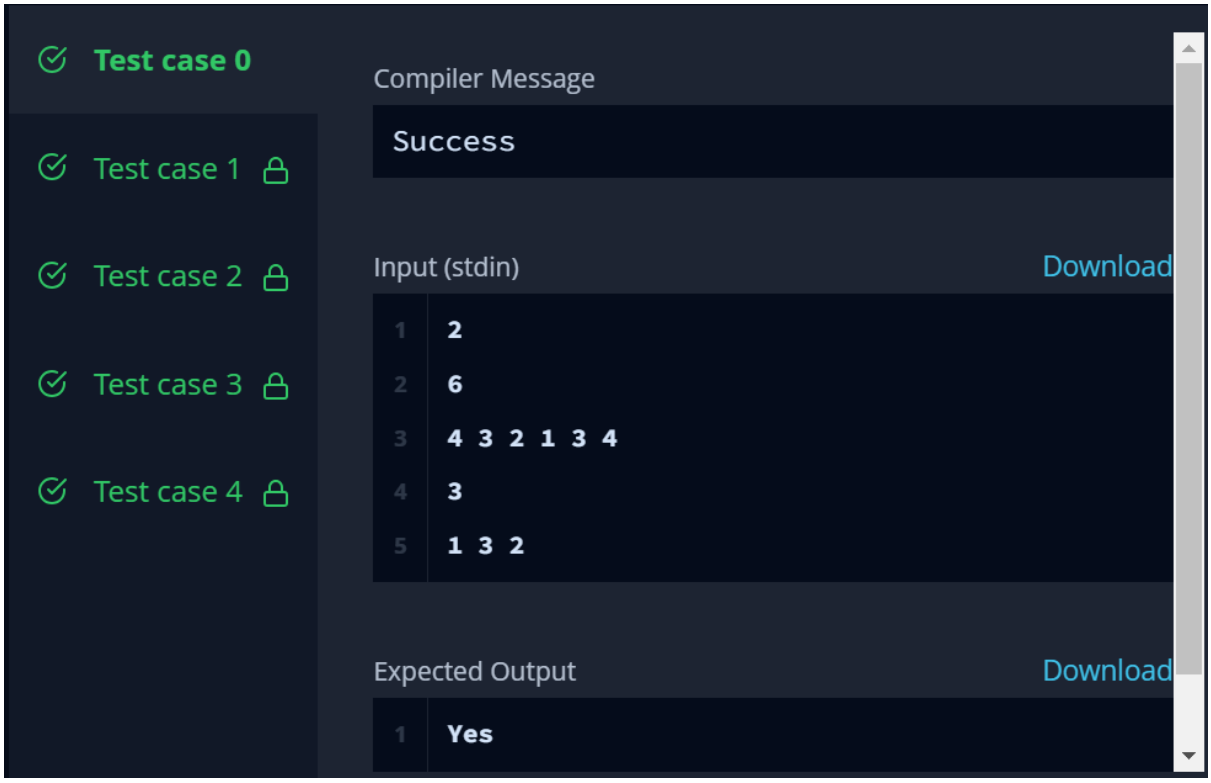
1.8. Company Logo

```
# importing the required modules
import math
import os
import random
import re
import sys
from collections import Counter

# Using __name__ variable
if __name__ == '__main__':

    # taking imput and then sorting
    s = sorted(input().strip())

    # finiding frequency
    s_counter =Counter(s).most_common()

    # using lambda function sort the items with frequencies in de
ceding order
    s_counter = sorted(s_counter, key=lambda x: (x[1] * -
1, x[0]))

    # printing the first three items
    for i in range(0, 3):
        print(s_counter[i][0], s_counter[i][1])
```



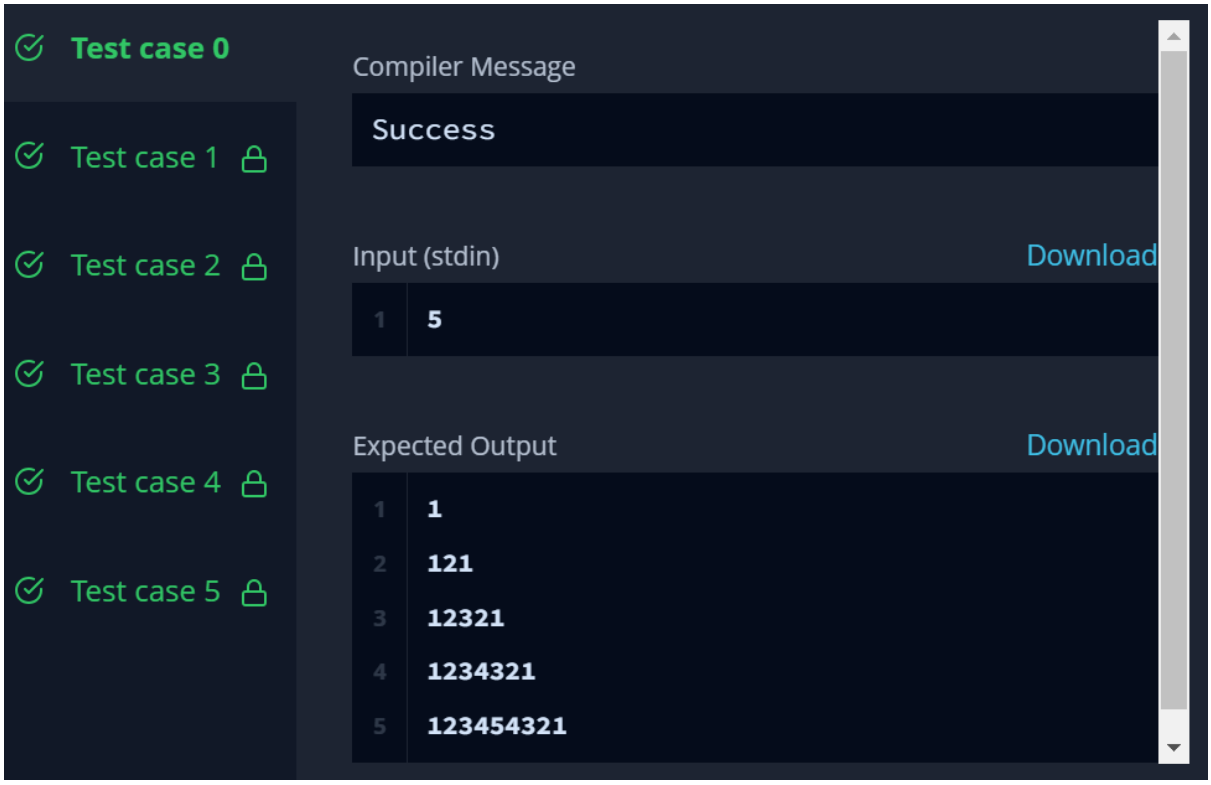
1.9. Piling Up!

```
t = int(input())

for _ in range(t):
    num, cubes = int(input()), list(map(int,input().split()))
    yes_no = "Yes"

    while len(cubes) > 1:
        if cubes[0] >= cubes[-1]:
            larger_num = cubes[0]
            cubes.pop(0)
        else:
            larger_num = cubes[-1]
            cubes.pop(-1)
        if larger_num < cubes[0] or larger_num < cubes[-1]:
            yes_no = 'No'
            break

    print(yes_no)
```



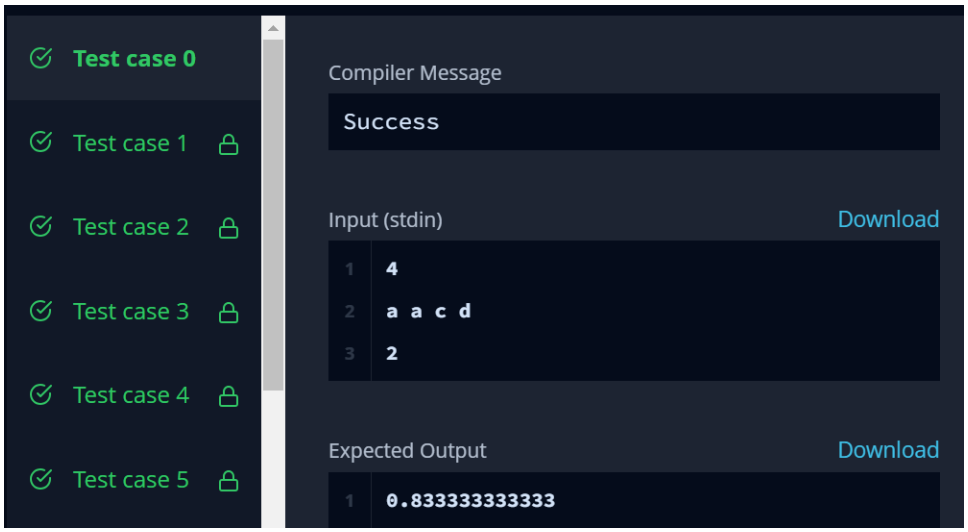
1.10. Triangle Quest 2

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



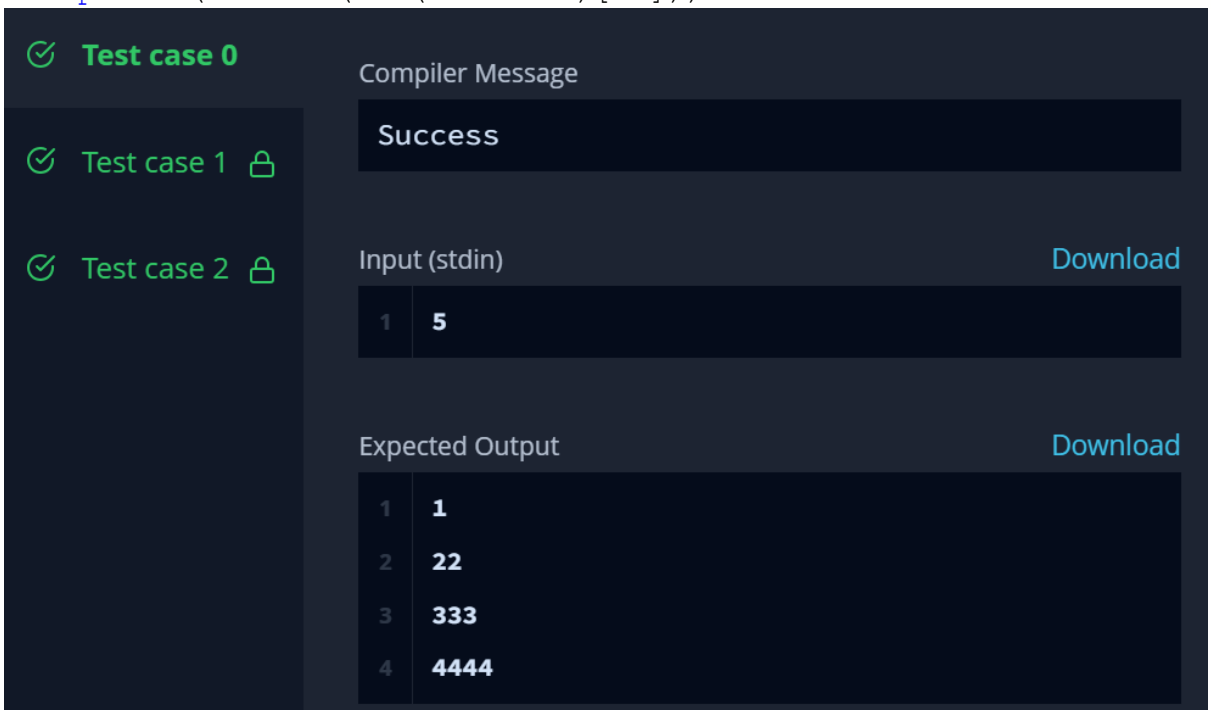
1.11. Iterables and Iterators

```
#!/usr/bin/env python3
import string
symbols = string.ascii_lowercase
from itertools import combinations
if __name__ == "__main__":
    n = int(input().strip())
    arr = list(map(str, input().strip().split(' ')))
    times = int(input().strip())
    cmbts = list(combinations(sorted(arr), times))
    print("{:.4f}".format(len(list(filter(lambda a: a[0] == 'a',
cmbts)))/(len(cmbts))))
```



1.12. Triangle Quest

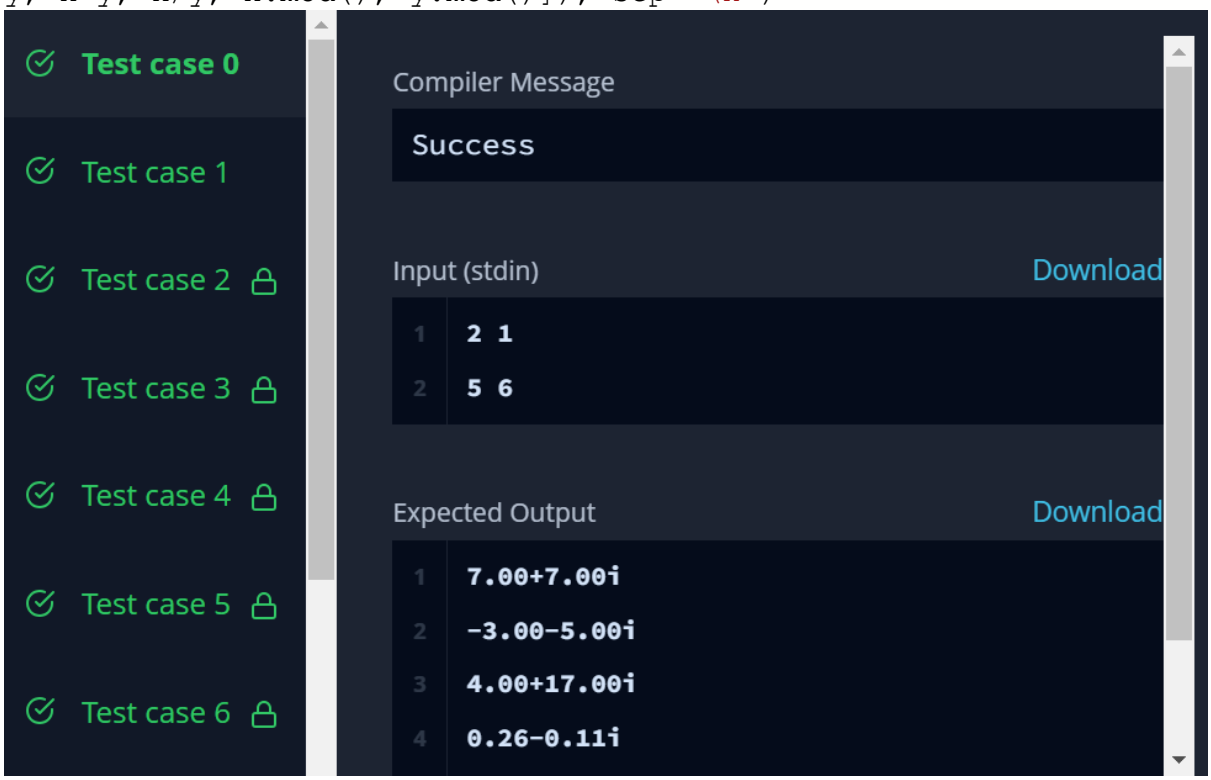
```
for i in range(1,int(input())):
    print (i * int(bin(2**i - 1)[2:]))
```



1.13. Classes: Dealing with Complex Numbers

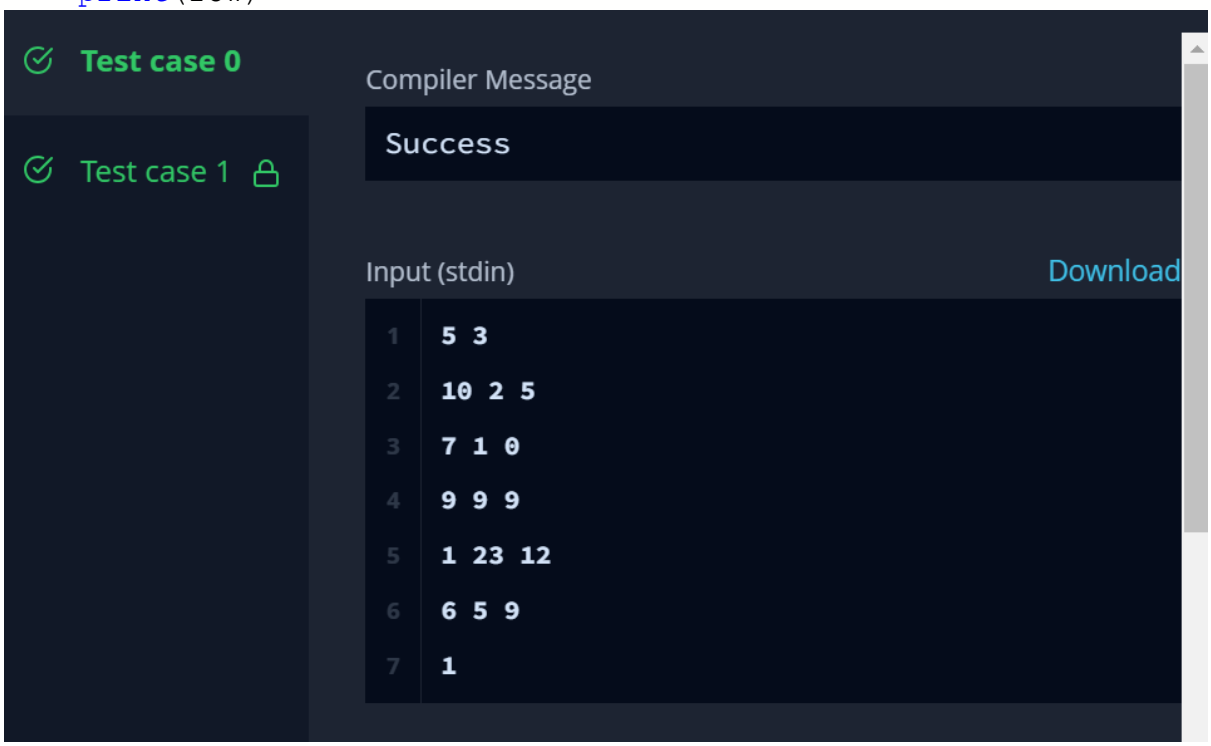
```
import math

class Complex(object):
    def __init__(self, real, img):
        self.real = real
        self.img = img

    def __add__(self, no):
        return Complex(self.real + no.real, self.img + no.img)

    def __sub__(self, no):
        return Complex(self.real - no.real, self.img - no.img)

    def __mul__(self, no):
        return Complex(self.real*no.real - self.img*no.img,
                       self.real*no.img + self.img*no.real)

    def __truediv__(self, no):
        return Complex((self.real*no.real + self.img*no.img)/(no.
real**2 + no.img**2),
                       (self.img*no.real - self.real*no.img)/(no.r
eal**2 + no.img**2))
```

```python
    def mod(self):
        return Complex((self.real**2 + self.img**2)**(1/2),
                       0)

    def __str__(self):
        if self.img == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            if self.img >= 0:
                result = "0.00+%.2fi" % (self.img)
            else:
                result = "0.00-%.2fi" % (abs(self.img))
        elif self.img > 0:
            result = "%.2f+%.2fi" % (self.real, self.img)
        else:
            result = "%.2f-%.2fi" % (self.real, abs(self.img))
        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')
```



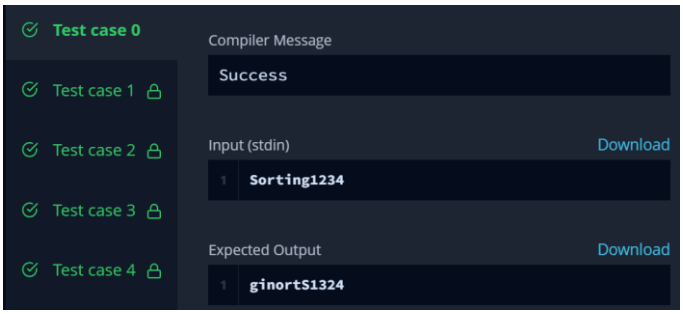
1.14. Athlete Sort

```
# initiailzing map function
N, M = map(int, input().split())
# taking for rows
rows = [input() for _ in range(N)]
# taking input from user
K = int(input())
# sorting using sorted function
for row in sorted(rows, key=lambda row: int(row.split()[K])):
    print(row)
```



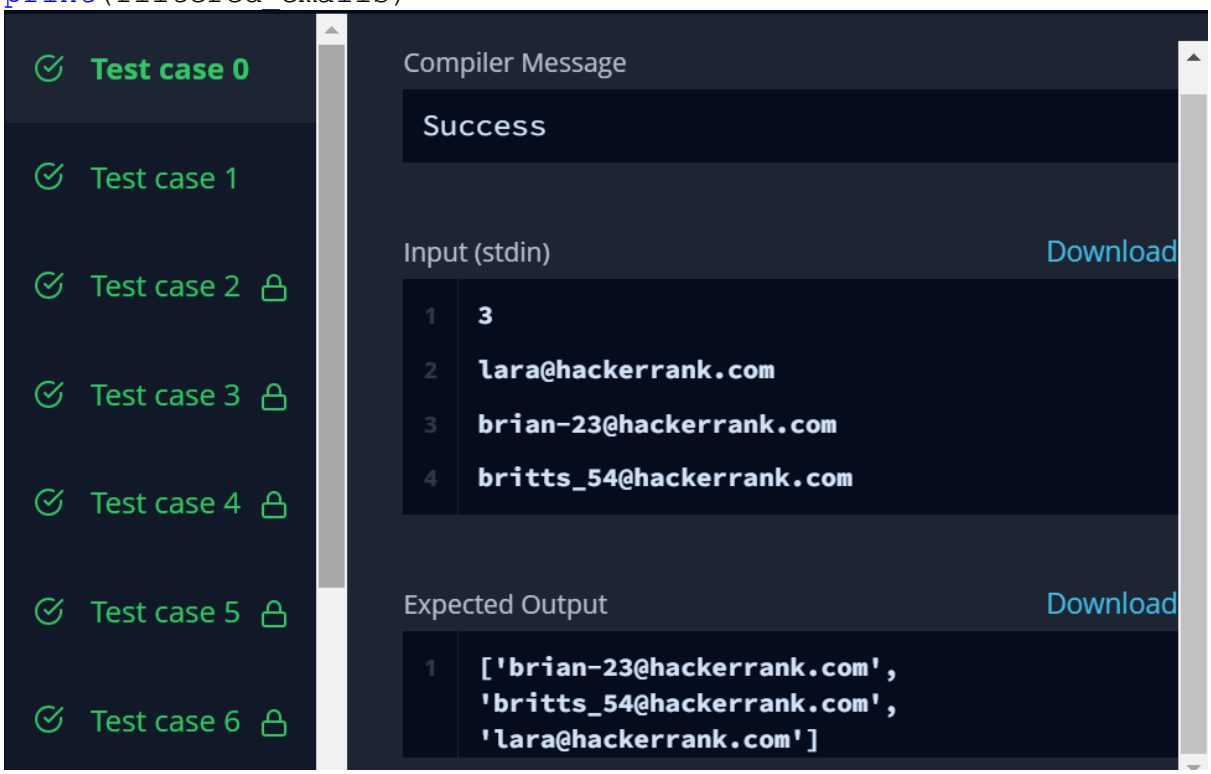
1.15. ginortS

```
#!/usr/bin/env python3
if __name__ == "__main__":
    string = input().strip()
    print(*sorted(string, key = lambda x: (-
x.islower(), x.isdigit() - x.isupper(), x in '02468', x)), sep=''
)
```



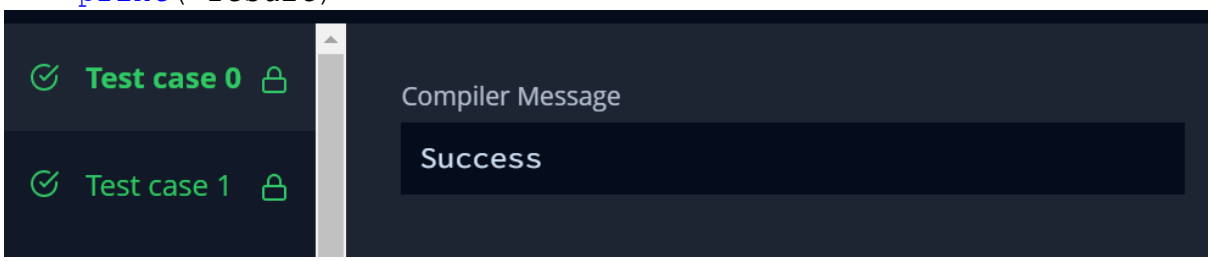
1.16. Validating Email Addresses with a Filter

```
# importing the module
import re

def fun(s):

    # using re.match function
    a = re.match(r'[a-zA-Z0-9_-]+@[a-zA-Z0-9]+\.[a-zA-
Z]{1,3}$',s)

    #returning the email adress
    return(a)

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)
```



1.17. 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)
```



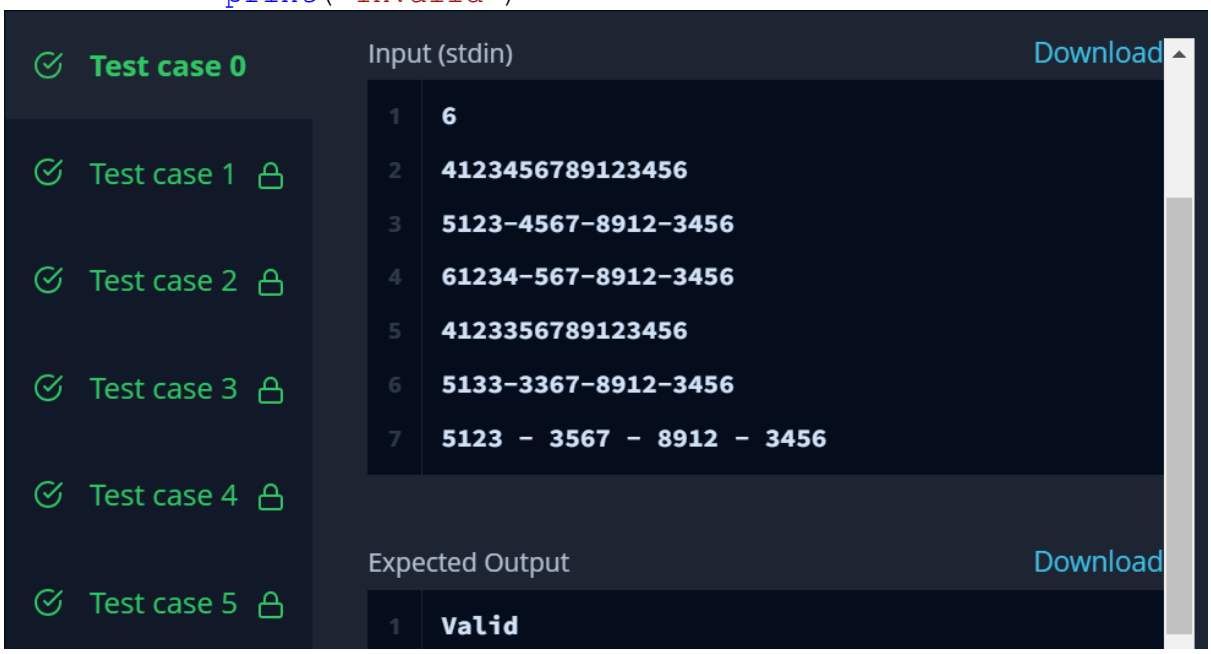
1.18. Regex Substitution

```
import re

def change(match):
    symb = match.group(0)

    if symb == "&&":
        return "and"
    elif symb == "||":
        return "or"

n = int(input().strip())
for _ in range(n):
    print(re.sub(r'(?<= )(&&|\||\|)(?= )', change, input()))
```

1.19. Validating Credit Card Numbers

```
import re
```

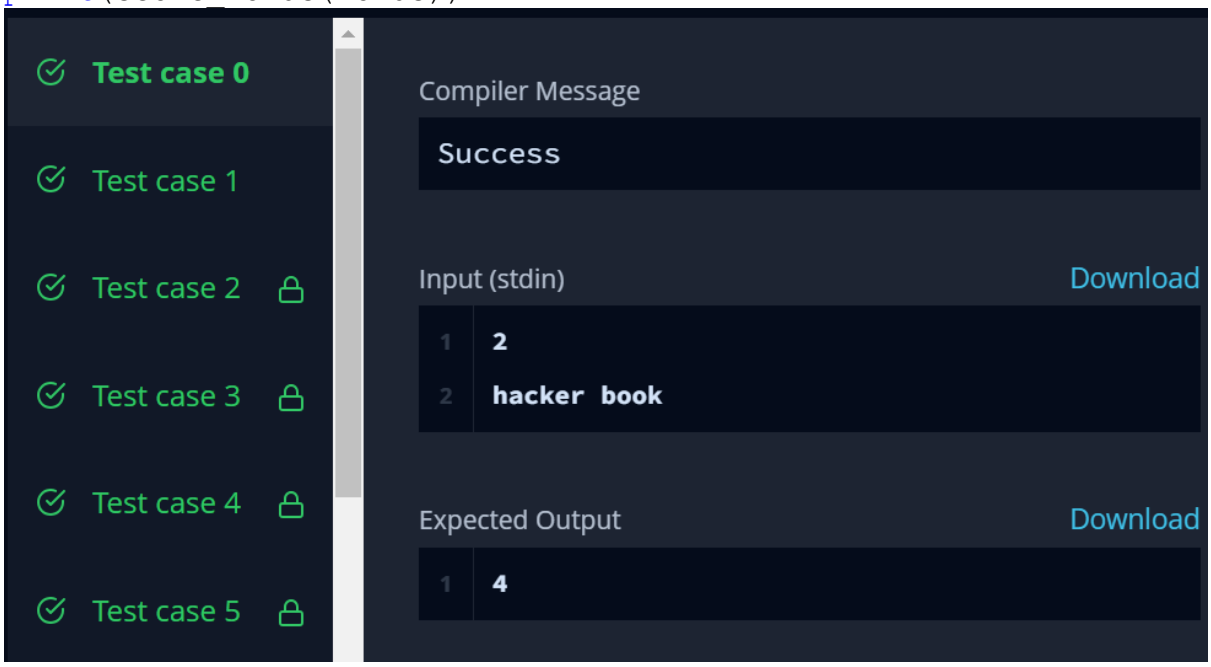
```python
if __name__ == "__main__":
    t = int(input().strip())

    for _ in range(t):
        num = "".join(input())
        if (re.match(r'^[456]', num) and
            (re.match(r'([\d]{4}-){3}[\d]{4}$', num) or
             re.match(r'[\d]{16}', num)) and
            not re.search(r'(\d)\1{3,}', num.replace("-", "")):
            print("Valid")
        else:
            print("Invalid")
```



1.20. Words Score

```python
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()
```

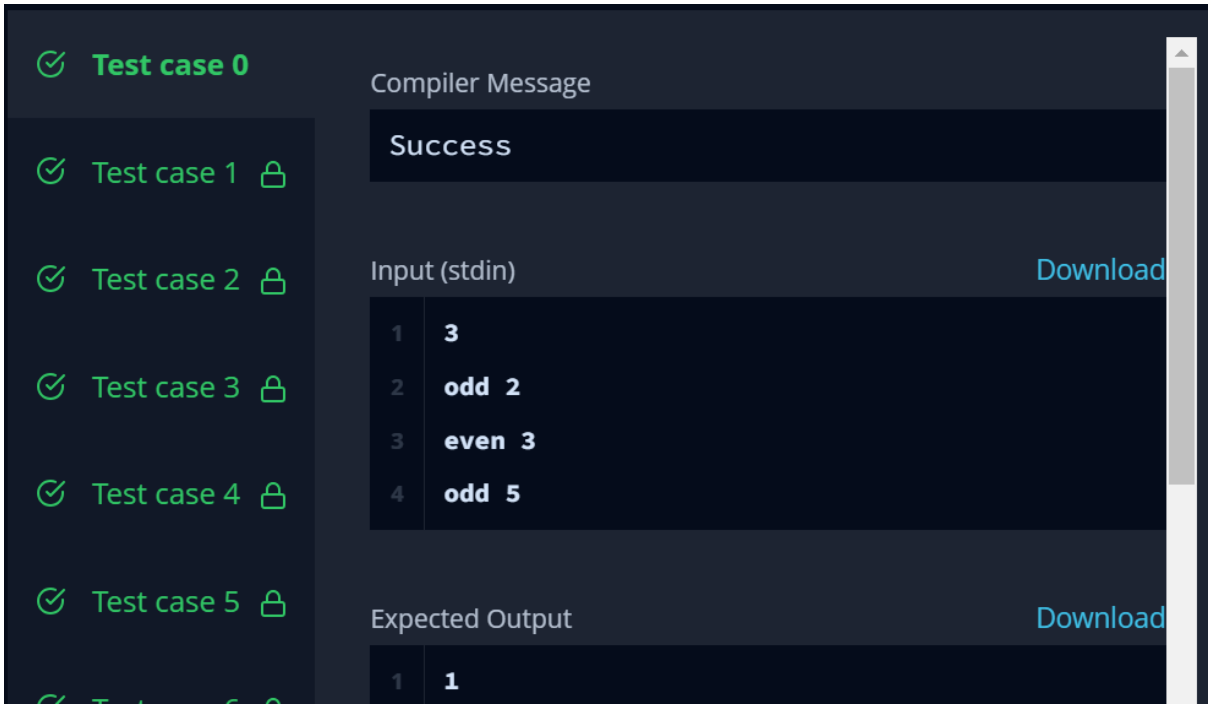
```
print(score_words(words))
```



1.21. 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 not stream:
        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())
```



1.22. Merge the Tools!

```
def merge_the_tools(string, k):
    block_cnt = len(string)//k
    output_t = []
    output_u = []

    #print("{}/{} = {}".format(len(string), k, block_len))
    for ind in range(0, len(string) - k + 1, k):
        output_t.append(string[ind:ind + k])

    for block in output_t:
        for char in block:
            char_count = block.count(char)
            if char_count > 1:
                block = block[::-1]
                block = block.replace(char, '', char_count - 1)
                block = block[::-1]
        output_u.append(block)

    print("\n".join(map(str, output_u)))

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

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

```
AABCAAADA
3
```

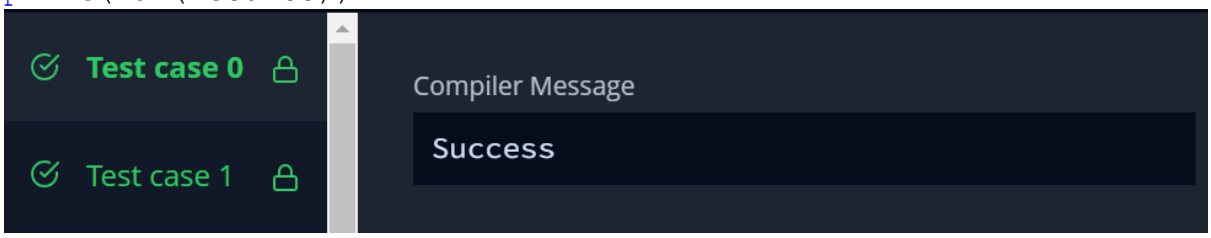
Expected Output Download

```
AB
CA
AD
```

2. Hard Challenges

2.1. Maximize It!

```
#!/usr/bin/env python3
from itertools import product
K,M = map(int,input().split())
N = (list(map(int, input().split())))[1:] for _ in range(K))
results = map(lambda x: sum(i**2 for i in x)%M, product(*N))
print(max(results))
```

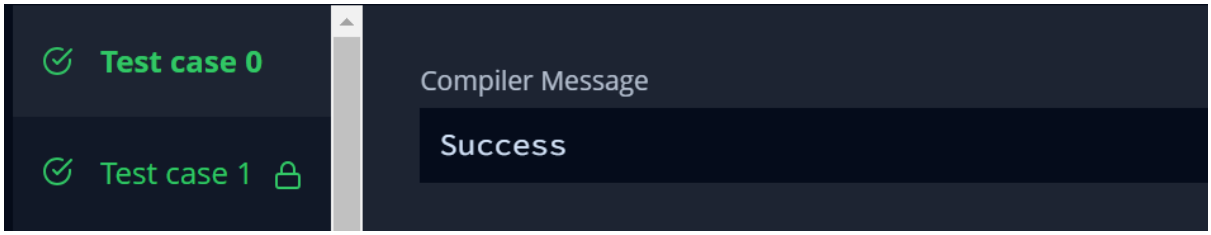


2.2. Validating Postal Codes

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

2.3. Matrix Script

```
import re
n, m = input().strip().split(' ')
n, m = [int(n), int(m)]
matrix = []
for _ in range(n):
    matrix_t = str(input())
    matrix.append(matrix_t)
complete = ""
for el in zip(*matrix):
    complete += "".join(el)
print(re.sub(r'(?<=\w)([^\w]+)(?=\w)', " ", complete))
```


Test case 0
Test case 1
Compiler Message
Success