

1. Medium Challenges

1.1. The Minion Game

```
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)
```

✓ 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

BANANA


Expected Output [Download](#)

1

Stuart 12

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))
```

 **Sample Test case 0**

Input (stdin)		Download
1	1990	

Your Output (stdout)	
1	False

Expected Output		Download
1	False	

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()
```


Sample Test case 0

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

Your Output (stdout)

1	25200
2	88200

Expected Output
[Download](#)

1	25200
2	88200

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/bc))))))
```

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°
---	-----

1.5. No Idea!

```
#!/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)
```

The screenshot shows a coding platform interface with a dark theme. On the left, a sidebar lists seven test cases, all marked as passed with green checkmarks. The main area on the right is divided into sections for compiler messages, input, and expected output.

Test Cases:

- 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	3 2
2	1 5 3
3	3 1
4	5 7

Expected Output:

1	1
---	---

1.6. Word Order

```
#!/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())))

```

The screenshot shows a coding platform interface. On the left, a sidebar lists test cases from 0 to 6, all marked as passed with green checkmarks. A tooltip labeled "Hidden test case" points to Test case 5. The main area displays the compiler message "Success". Below it, the input (stdin) is shown as a table with 5 rows: 1: 4, 2: bcdef, 3: abcdefg, 4: bcde, 5: bcdef. A "Download" link is next to the input. Below the input, the expected output is shown as a table with 1 row: 1: 3. A "Download" link is next to the expected output.

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=' ')

```

The screenshot shows a coding platform interface. On the left, a sidebar lists test cases from 0 to 4, all marked as passed with green checkmarks. The main area displays the compiler message "Success". Below it, the input (stdin) is shown as a table with 1 row: 1: 1222311. A "Download" link is next to the input. Below the input, the expected output is shown as a table with 1 row: 1: (1, 1) (3, 2) (1, 3) (2, 1). A "Download" link is next to the expected output.

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 input 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])
```

✓ 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 aabbbccde
```

Expected Output

[Download](#)

```
1 b 3
2 a 2
3 c 2
```

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)
```

The screenshot shows a coding platform interface with a dark theme. On the left, there is a sidebar with five test cases, each marked with a green checkmark and a lock icon: 'Test case 0', 'Test case 1', 'Test case 2', 'Test case 3', and 'Test case 4'. The main area is divided into two sections. The top section, titled 'Compiler Message', shows a 'Success' message. The bottom section, titled 'Input (stdin)', displays the input for the first test case: a list of numbers [2, 6, 4, 3, 2, 1, 3, 4, 3, 1, 3, 2] across five lines. To the right of the input section is a 'Download' button. Below the input section is another section titled 'Expected Output', which shows the output 'Yes' for the first test case. To the right of the expected output section is another 'Download' button.

Test Case	Input (stdin)	Expected Output
1	2 6 4 3 2 1 3 4 3 1 3 2	Yes

1.10. Triangle Quest 2

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


✓ 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	5
---	---

Expected Output [Download](#)

1	1
2	121
3	12321
4	1234321
5	123454321

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))))
```

✓ 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	4
2	a a c d
3	2

Expected Output [Download](#)

1	0.833333333333
---	----------------

1.12. Triangle Quest

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

✓ Test case 0

✓ Test case 1

✓ Test case 2

Compiler Message

Success

Input (stdin) [Download](#)

1	5
---	---

Expected Output [Download](#)

1	1
2	22
3	333
4	4444

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))
```

✓ 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](#)

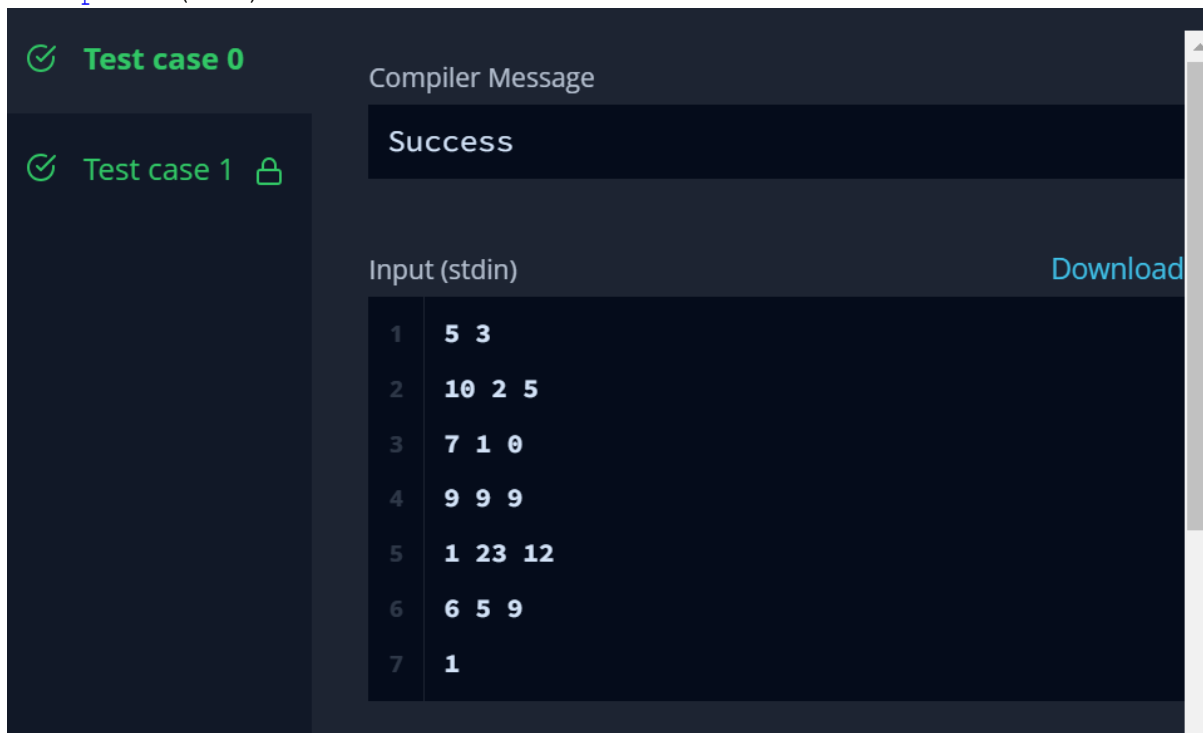
1	2 1
2	5 6

Expected Output [Download](#)

1	7.00+7.00i
2	-3.00-5.00i
3	4.00+17.00i
4	0.26-0.11i

1.14. Athlete Sort

```
# initiallizing 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)
```



Test case 0

Test case 1

Compiler Message

Success

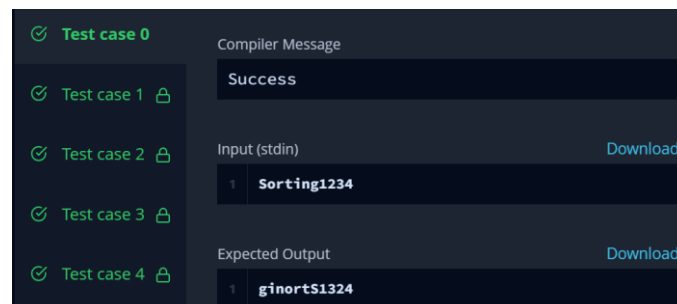
Input (stdin)

1	5	3
2	10	2 5
3	7	1 0
4	9	9 9
5	1	23 12
6	6	5 9
7	1	

Download

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='')
)
```



Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Compiler Message

Success

Input (stdin)

1	Sorting1234
---	-------------

Download

Expected Output

1	ginortS1324
---	-------------

Download

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 address
    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)
```

The screenshot displays the HackerRank interface for the 'Validating Email Addresses with a Filter' problem. On the left, a sidebar lists seven test cases, all marked as 'Test case 0' with a green checkmark. The main area is divided into two sections: 'Compiler Message' and 'Input (stdin)'. The 'Compiler Message' section shows a 'Success' status. The 'Input (stdin)' section shows the input data for the test case: 3, followed by three email addresses: lara@hackerrank.com, brian-23@hackerrank.com, and britts_54@hackerrank.com. Below this, the 'Expected Output' section shows the expected result: ['brian-23@hackerrank.com', 'britts_54@hackerrank.com', 'lara@hackerrank.com']. Each section has a 'Download' link.

Test Case	Status
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

Compiler Message

Success

Input (stdin)

Download

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

Expected Output

Download

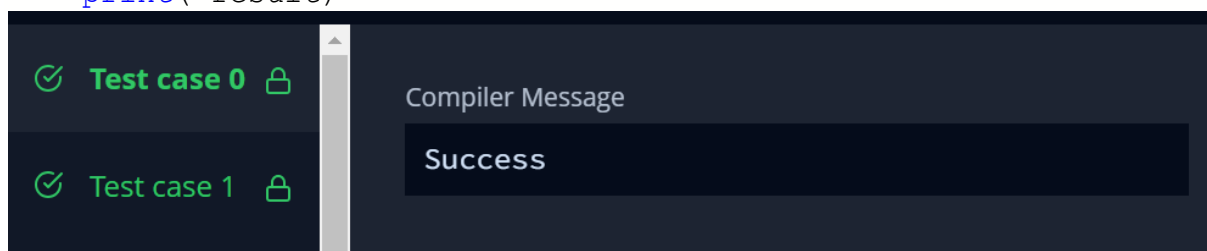
1	['brian-23@hackerrank.com', 'britts_54@hackerrank.com', 'lara@hackerrank.com']
---	--

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)
```



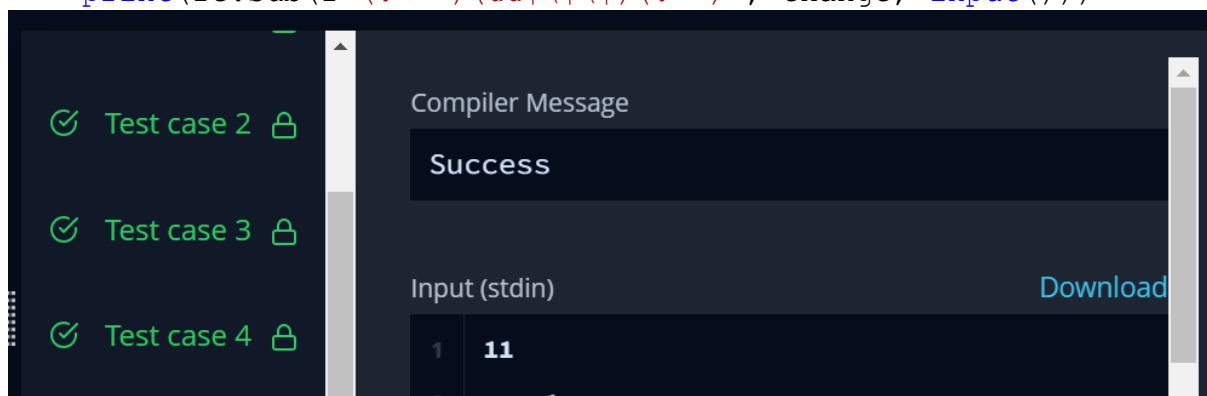
1.18. Regex Substitution

```
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
```

```

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")

```

Test case 0	Input (stdin)	Download
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	
Test case 1	Expected Output	Download
1	Valid	

1.20. 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))
```

The screenshot shows a code editor interface with a dark theme. On the left, there is a sidebar with a list of test cases, each preceded by a green checkmark icon. The test cases are labeled 'Test case 0' through 'Test case 5'. 'Test case 0' is highlighted in green. To the right of the sidebar, the main editor area displays the 'Compiler Message' section, which shows a 'Success' message. Below this, the 'Input (stdin)' section shows two lines of input: '2' and 'hacker book'. To the right of the input section is a 'Download' link. Below the input section, the 'Expected Output' section shows a single line of output: '4'. To the right of the expected output section is another 'Download' link.

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())

```

The screenshot shows a coding platform interface with a dark theme. On the left, there is a sidebar with a list of test cases, each preceded by a green checkmark and a lock icon. The test cases are labeled 'Test case 0' through 'Test case 5'. The main area on the right is titled 'Compiler Message' and shows a 'Success' message. Below this, there is a section for 'Input (stdin)' with a 'Download' link. The input is displayed in a table with four rows: row 1 contains '3', row 2 contains 'odd 2', row 3 contains 'even 3', and row 4 contains 'odd 5'. Below the input section, there is a section for 'Expected Output' with a 'Download' link. The expected output is displayed in a table with one row: row 1 contains '1'.

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](#)

1	AABCAAADA
2	3

Expected Output [Download](#)

1	AB
2	CA
3	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))
```

✓ Test case 0	Compiler Message
✓ Test case 1	Success

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)
```

✓ Test case 0	Compiler Message
✓ Test case 1	Success
✓ Test case 2	Input (stdin) Download
✓ Test case 3	1 110000
✓ Test case 4	Expected Output Download
	1 False

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