Artificial Intelligence CSE-860B

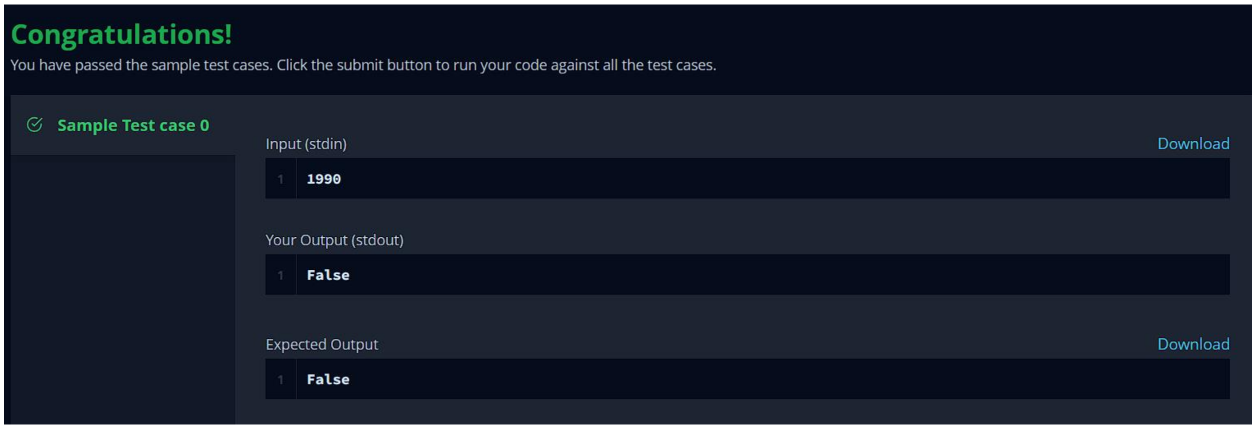
Assignment No. 3

This assignment contains python codes and screenshots of MEDIUM and HARD challenges of HACKERRANK (https://www.hackerrank.com/).

MEDIUM CHALLENGES

1. *Write a Function*

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



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)

```
1990
```

Your Output (stdout)

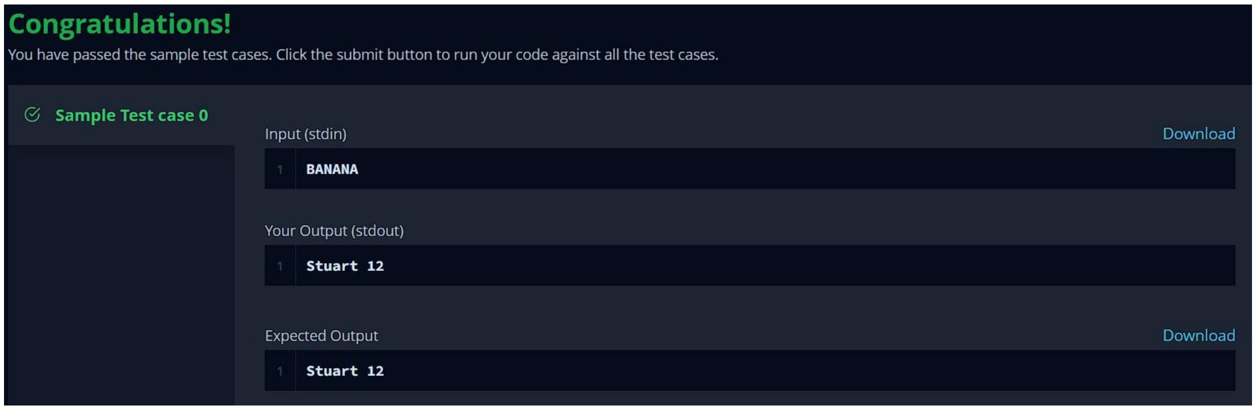
```
False
```

Expected Output

```
False
```

2. *The Minion Game*

```
def minion_game(string):
    vowels = 'AEIOU'
    kevin_score, stuart_score = 0, 0
    length = len(string)
    for i in range(length):
        if string[i] in vowels:
            kevin_score += length - i
        else:
            stuart_score += length - i
    if kevin_score > stuart_score:
        print(f"Kevin {kevin_score}")
    elif kevin_score < stuart_score:
        print(f"Stuart {stuart_score}")
    else:
        print("Draw")
if __name__ == '__main__':
    s = input()
    minion_game(s)
```


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)

```
BANANA
```

Your Output (stdout)

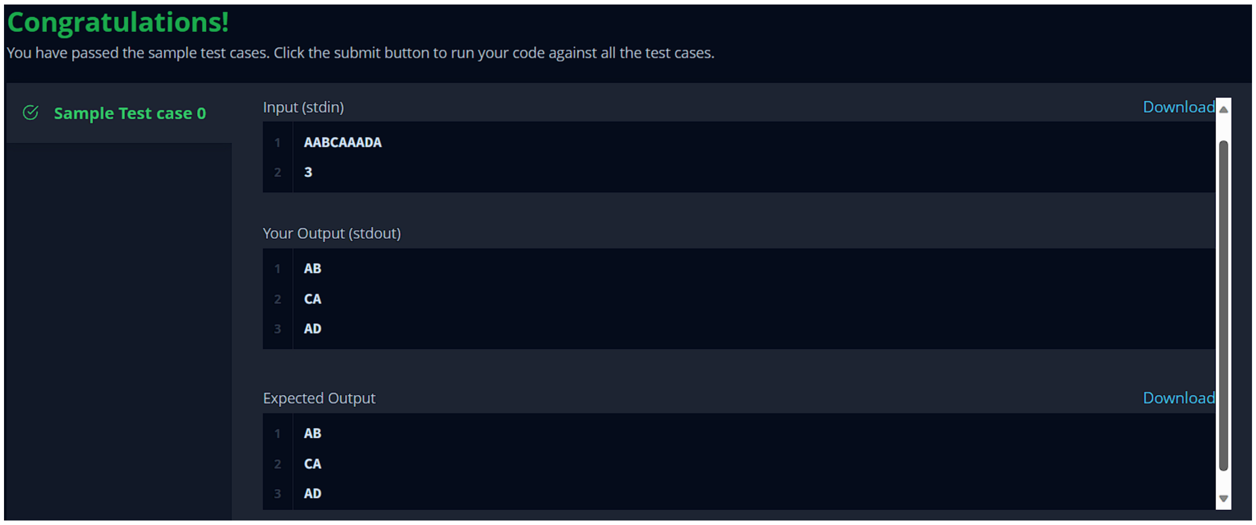
```
Stuart 12
```

Expected Output

```
Stuart 12
```

3. *Merge The Tools*

```
def merge_the_tools(string, k):
    n = len(string)
    for i in range(0, n, k):
        sub_string = string[i:i + k]
        unique_chars = []
        for ch in sub_string:
            if ch not in unique_chars:
                unique_chars.append(ch)
        print(''.join(unique_chars))
if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```


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

Input (stdin)

```
AABCAAADA
3
```

Your Output (stdout)

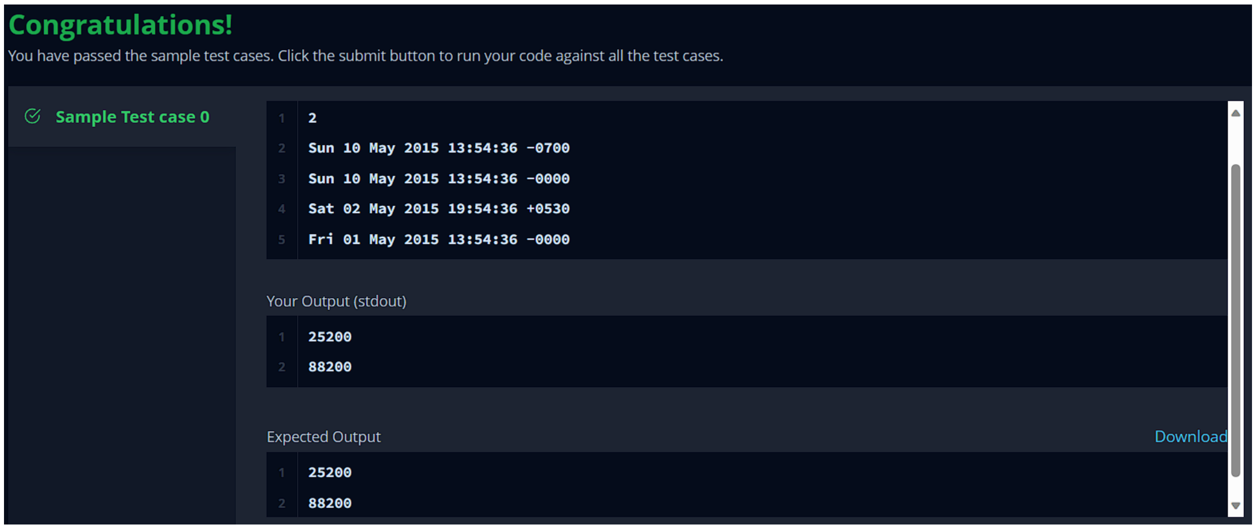
```
AB
CA
AD
```

Expected Output

```
AB
CA
AD
```

4. *Time Delta*

```
import math
import os
import random
import re
import sys
from datetime import datetime
def time_delta(t1, t2):
    format_str = '%a %d %b %Y %H:%M:%S %z'
    time1 = datetime.strptime(t1, format_str)
    time2 = datetime.strptime(t2, format_str)
    time_diff = abs(time1 - time2)
    return str(int(time_diff.total_seconds()))
if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    t = int(input())
    for t_itr in range(t):
        t1 = input()
        t2 = input()
        delta = time_delta(t1, t2)
        fptr.write(delta + '\n')
    fptr.close()
```


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

```
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)

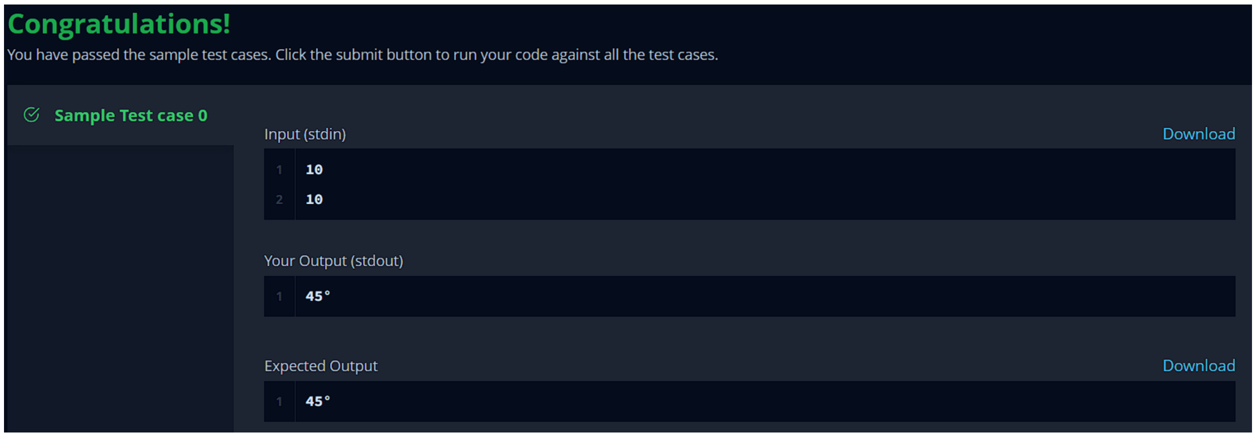
```
25200
88200
```

Expected Output

```
25200
88200
```

5. *Find Angle MBC*

```
import math
def find_angle(ab, bc):
    angle_rad = math.atan2(ab, bc)
    angle_deg = math.degrees(angle_rad)
    return round(angle_deg)
side_ab = int(input())
side_bc = int(input())
if side_ab > 0 and side_ab <= 100 and side_bc > 0 and side_bc <= 10
0:
    angle = find_angle(side_ab, side_bc)
    degree_sign = chr(176)
    print(f'{angle}{degree_sign}')
else:
    print(f'wrong input')
```


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)

```
10
10
```

Your Output (stdout)

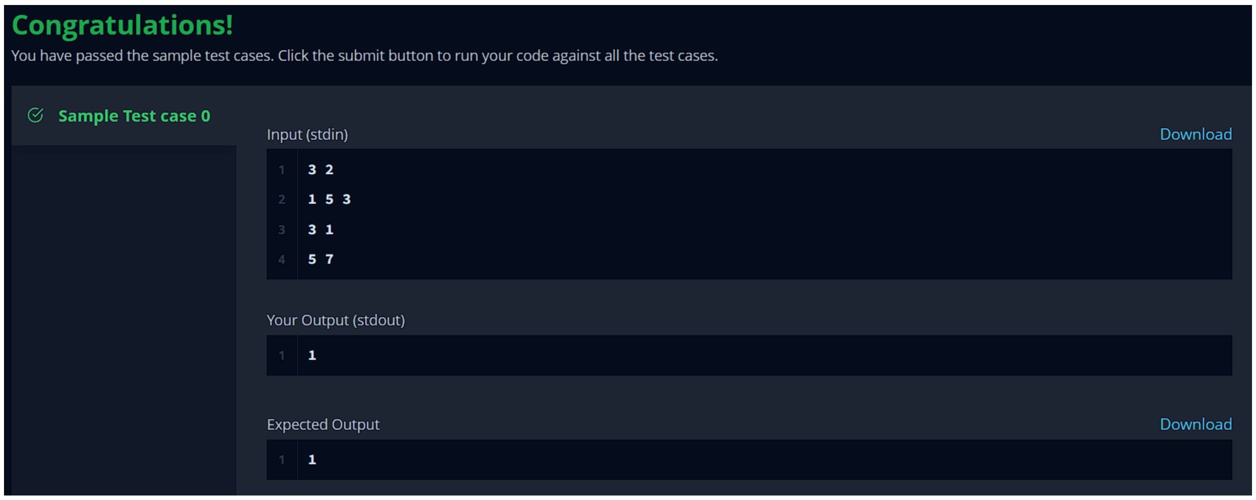
```
45°
```

Expected Output

```
45°
```

6. No Idea!

```
def calculate_happiness(array, set_like, set_dislike):
    happiness = 0
    for num in array:
        if num in set_like:
            happiness += 1
        elif num in set_dislike:
            happiness -= 1
    return happiness
n, m = map(int, input().split())
array = list(map(int, input().split()))
set_like = set(map(int, input().split()))
set_dislike = set(map(int, input().split()))
result = calculate_happiness(array, set_like, set_dislike)
print(result)
```


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)

```
3 2
1 5 3
3 1
5 7
```

Your Output (stdout)

```
1
```

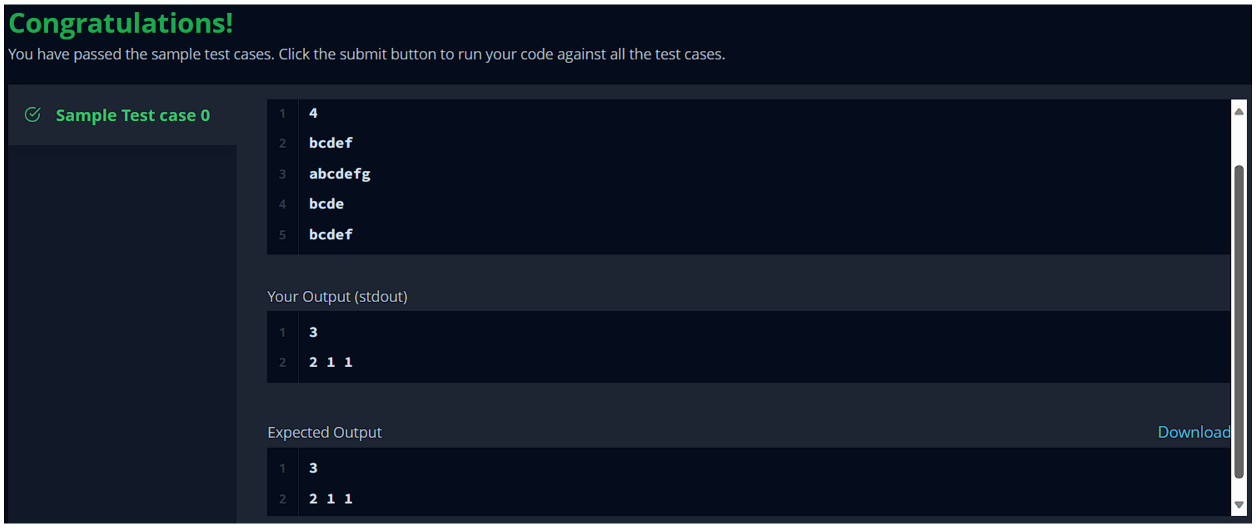
Expected Output

```
1
```

7. Word Order

```
from collections import OrderedDict
def count_word_occurrences(words):
    word_count = OrderedDict()
    for word in words:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1
    return word_count
n = int(input())
input_words = [input().strip() for _ in range(n)]
```

```
word_occurrences = count_word_occurrences(input_words)
distinct_words = len(word_occurrences)
occurrences = word_occurrences.values()
print(distinct_words)
print(*occurrences)
```


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

```
4
bcdef
abcdefg
bcde
bcdef
```

Your Output (stdout)

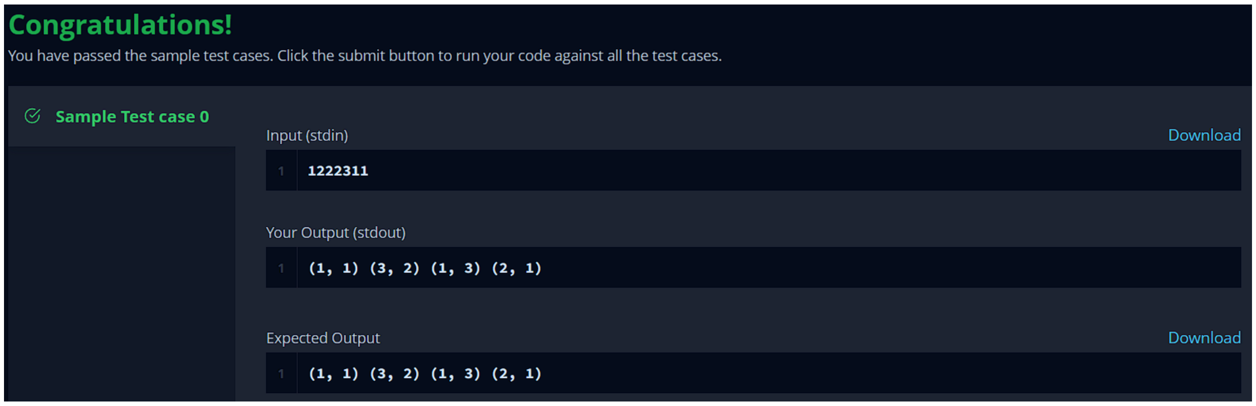
```
3
2 1 1
```

Expected Output

```
3
2 1 1
```

8. Compress The String

```
from itertools import groupby
def compress_string(s):
    compressed = [(len(list(group)), int(key)) for key, group in gr
oupby(s)]
    return compressed
if __name__ == '__main__':
    string = input().strip()
    compressed_string = compress_string(string)
    for count, num in compressed_string:
        print(f"({count}, {num})", end=' ')
```


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)

```
1222311
```

Your Output (stdout)

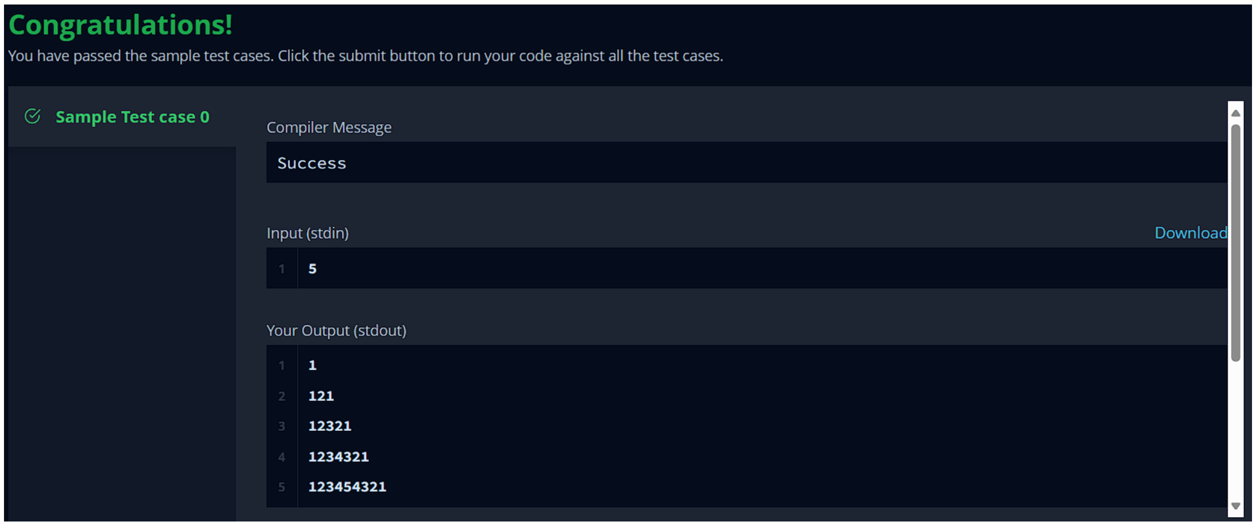
```
(1, 1) (3, 2) (1, 3) (2, 1)
```

Expected Output

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

9. Triangle Quest 2

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



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

Compiler Message

```
Success
```

Input (stdin)

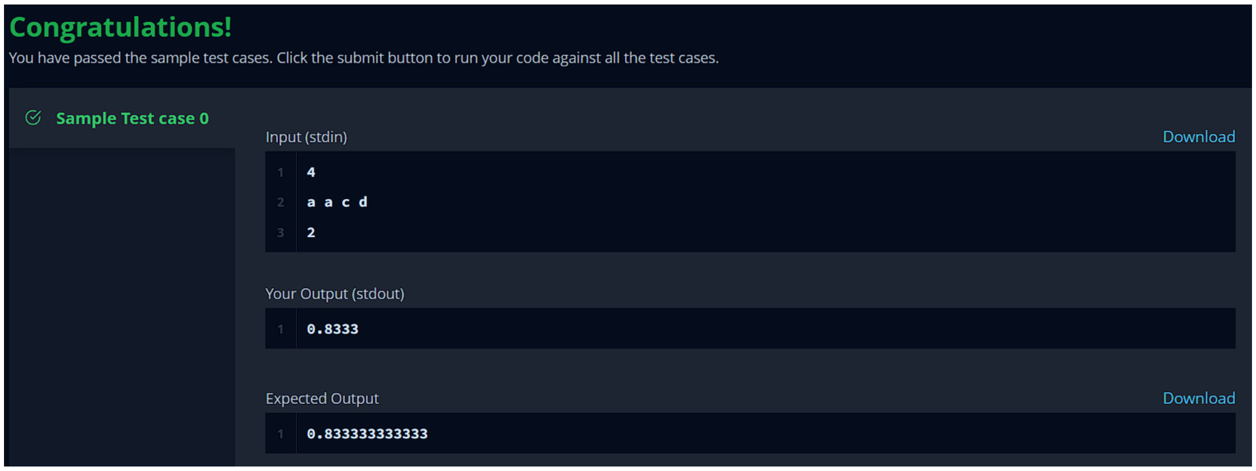
```
5
```

Your Output (stdout)

```
1
121
12321
1234321
123454321
```

10. Iterables and Iterators

```
from itertools import combinations
def probability_of_char(n, letters, k):
    total_combinations = list(combinations(letters, k))
    count = sum('a' in combo for combo in total_combinations)
    probability = count / len(total_combinations)
    return probability
if __name__ == "__main__":
    N = int(input())
    letters = input().split()
    K = int(input())
    probability = probability_of_char(N, letters, K)
    print(f"{probability:.4f}")
```



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)

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

Your Output (stdout)

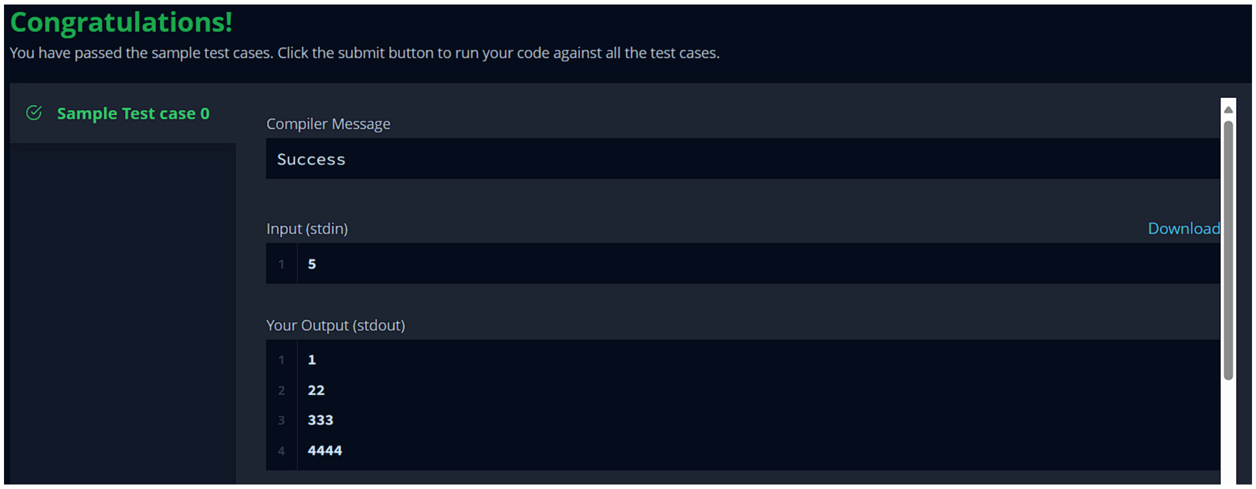
```
0.8333
```

Expected Output

```
0.833333333333
```

11. Triangle Quest

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



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

Compiler Message

```
Success
```

Input (stdin)

```
5
```

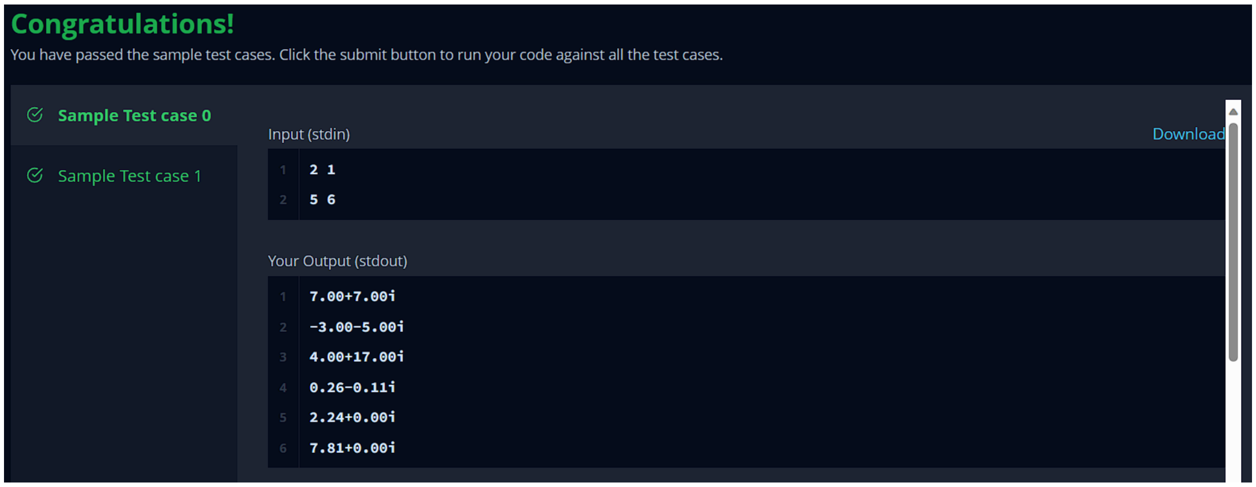
Your Output (stdout)

```
1
22
333
4444
```

12. Classes: Dealing with Complex Numbers

```
import math
class Complex(object):
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary
    def __add__(self, other):
        return Complex(self.real + other.real, self.imaginary + oth
er.imaginary)
    def __sub__(self, other):
        return Complex(self.real - other.real, self.imaginary - oth
er.imaginary)
```

```
    def __mul__(self, other):
        real = self.real * other.real - self.imaginary * other.imag
inary
        imaginary = self.real * other.imaginary + self.imaginary *
other.real
        return Complex(real, imaginary)
    def __truediv__(self, other):
        denominator = other.real**2 + other.imaginary**2
        real = (self.real * other.real + self.imaginary * other.ima
ginary) / denominator
        imaginary = (self.imaginary * other.real - self.real * othe
r.imaginary) / denominator
        return Complex(real, imaginary)
    def mod(self):
        return Complex(math.sqrt(self.real**2 + self.imaginary**2),
 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')
```


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

Input (stdin)

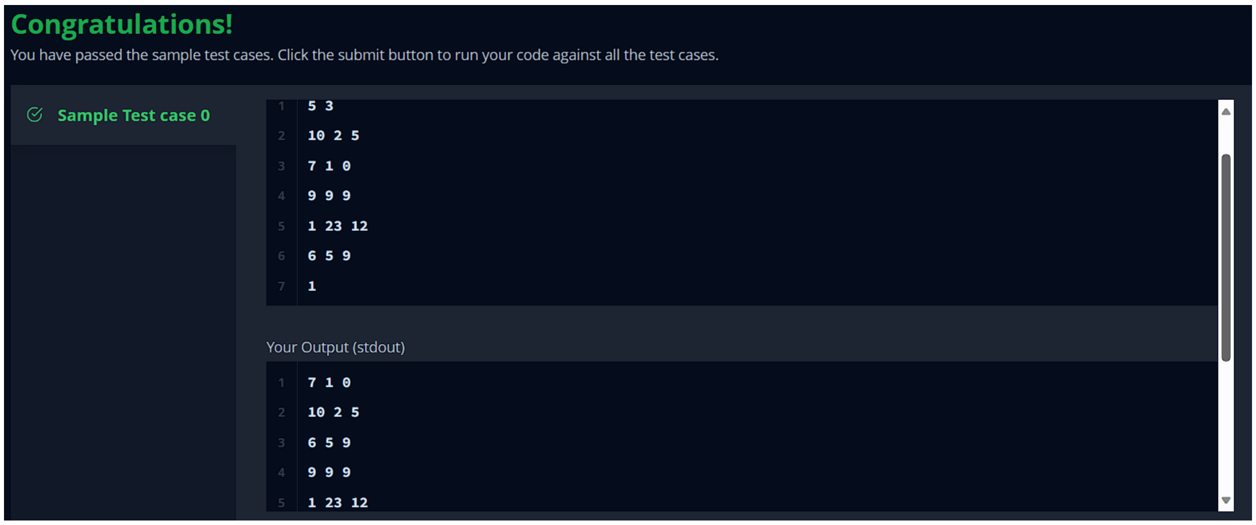
```
2 1
5 6
```

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
```

13. Athlete Sort

```
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())
    sorted_athletes = sorted(arr, key=lambda x: x[k])
    for athlete in sorted_athletes:
        print(*athlete)
```



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

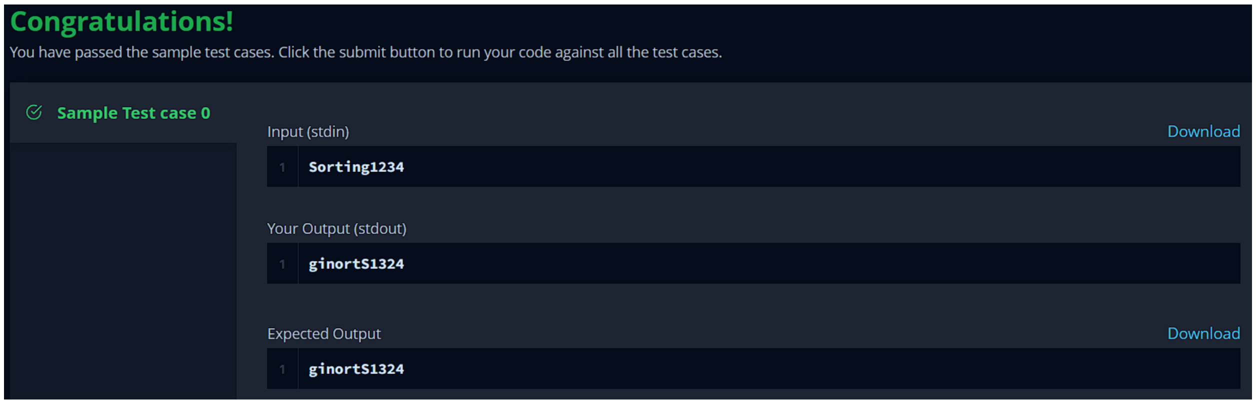
```
5 3
10 2 5
7 1 0
9 9 9
1 23 12
6 5 9
1
```

Your Output (stdout)

```
7 1 0
10 2 5
6 5 9
9 9 9
1 23 12
```

14. Ginorts

```
s = input()
sorted_string = sorted(s, key=lambda x: (x.isdigit(), x.isdigit() a
nd int(x) % 2 == 0, x.isupper(), x))
print(''.join(sorted_string))
```



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)

```
Sorting1234
```

Your Output (stdout)

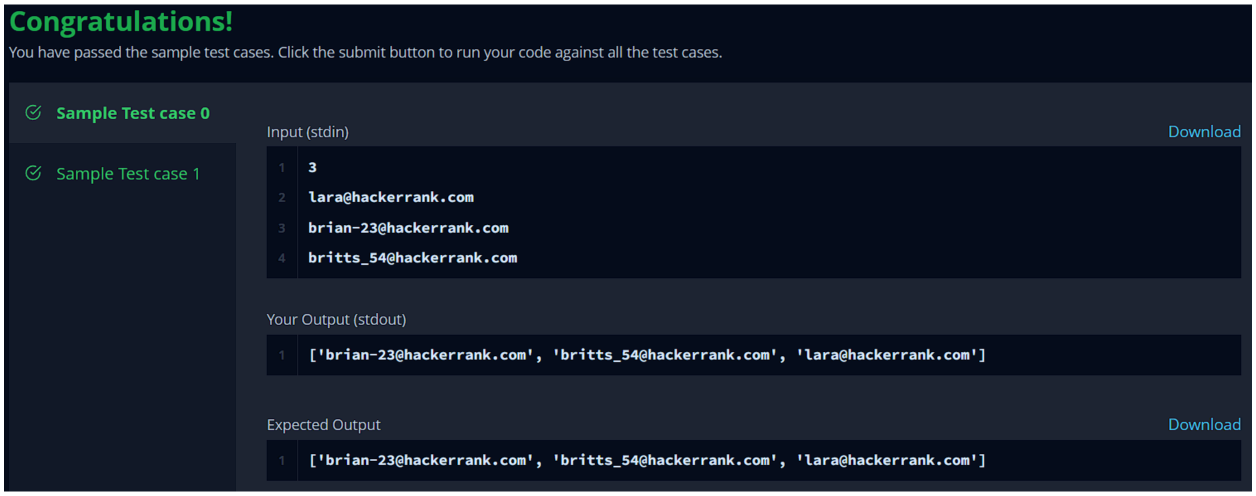
```
ginortS1324
```

Expected Output

```
ginortS1324
```

15. Validating email addresses with a filter

```
import re
def fun(emails):
    pattern = r'^[\w-]+@[a-zA-Z0-9]+\.\w{1,3}$'
    return re.match(pattern, 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)
```


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

Input (stdin)

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

Your Output (stdout)

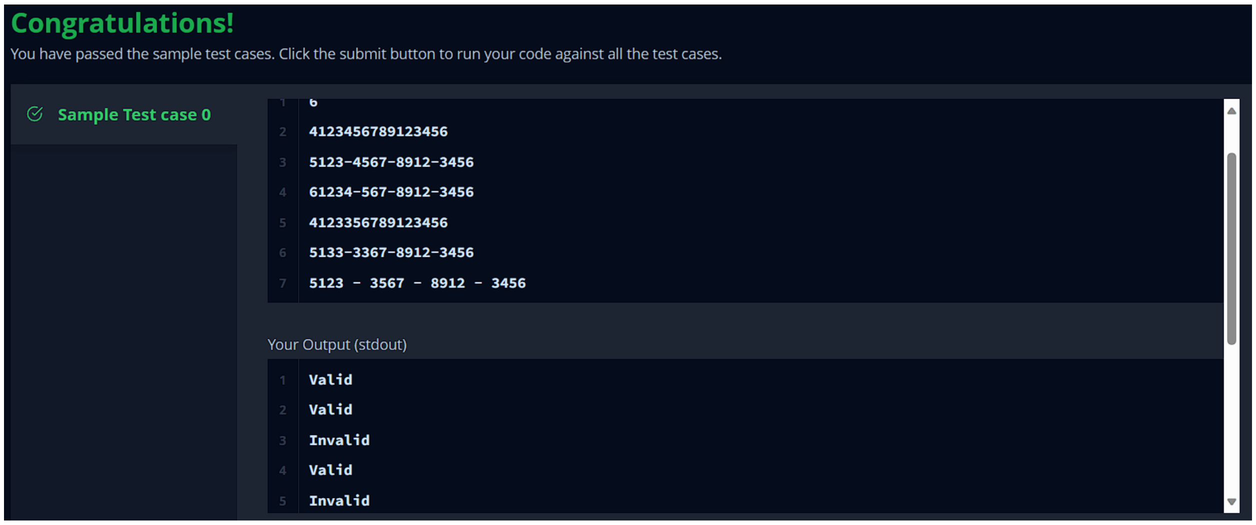
```
['brian-23@hackerrank.com', 'britts_54@hackerrank.com', 'lara@hackerrank.com']
```

Expected Output

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

16. Validating credit cards

```
import re
def validate_credit_cards(n, cards):
    for card in cards:
        if re.match(r'^[456]\d{3}(-
?\d{4}){3}$', card) and not re.search(r'(\d)\1{3,}', card.replace('
-', '')):
            print("Valid" if re.search(r'^([456]\d{15}|[456]\d{3}-
\d{4}-\d{4}-\d{4})$', card) else "Invalid")
        else:
            print("Invalid")
if __name__ == "__main__":
    n = int(input())
    credit_cards = [input() for _ in range(n)]
    validate_credit_cards(n, credit_cards)
```



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

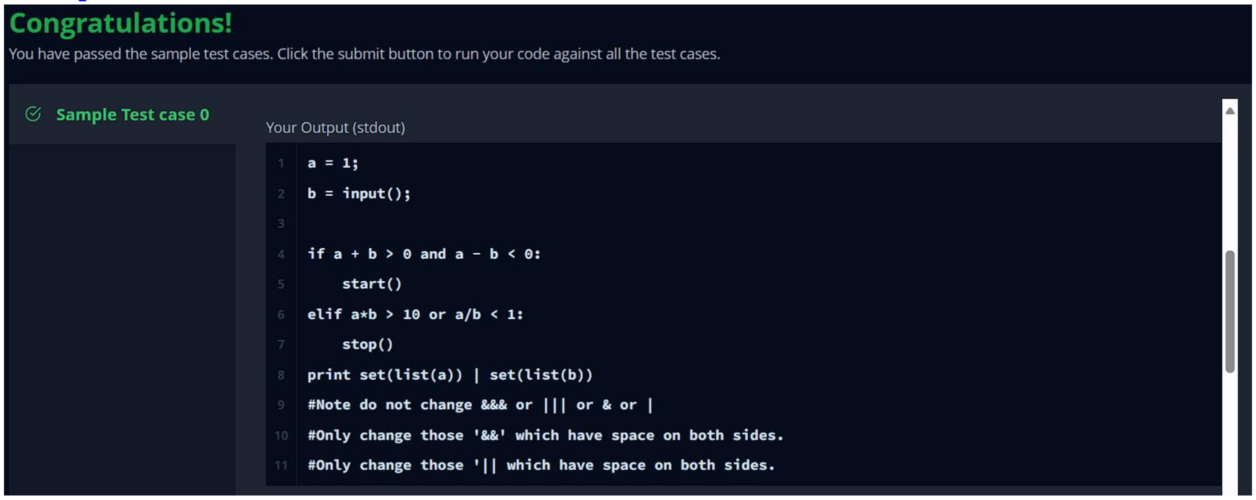
```
6
4123456789123456
5123-4567-8912-3456
61234-567-8912-3456
4123356789123456
5133-3367-8912-3456
5123 - 3567 - 8912 - 3456
```

Your Output (stdout)

```
Valid
Valid
Invalid
Valid
Invalid
```

17. Regex Substitution

```
import re
def substitution(text):
    pattern_and = r'(?<= )&&(?= )'
    pattern_or = r'(?<= )\|\|(?= )'
    text = re.sub(pattern_and, 'and', text)
    text = re.sub(pattern_or, 'or', text)
    return text
if __name__ == '__main__':
    n = int(input())
    lines = [input() for _ in range(n)]
    text = '\n'.join(lines)
    result = substitution(text)
    print(result)
```



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

Your Output (stdout)

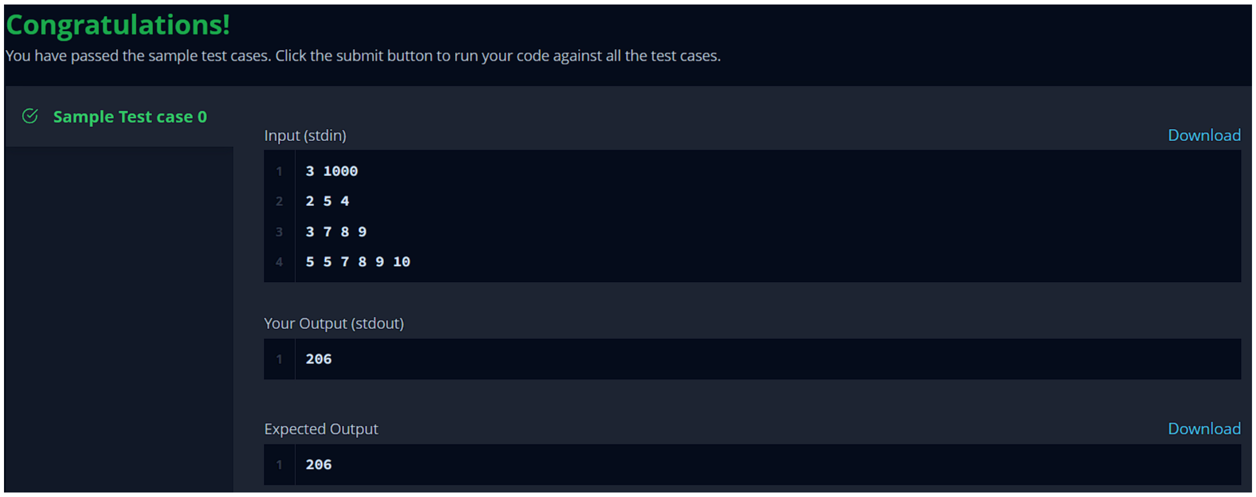
```
a = 1;
b = input();

if a + b > 0 and a - b < 0:
    start()
elif a*b > 10 or a/b < 1:
    stop()
print set(list(a)) | set(list(b))
#Note do not change &&& or ||| or & or |
#Only change those '&&' which have space on both sides.
#Only change those '|| which have space on both sides.
```

HARD CHALLENGES

1. *Maximize It!*

```
from itertools import product
K, M = map(int, input().split())
lists = []
for _ in range(K):
    vals = list(map(int, input().split()))[1:]
    lists.append(vals)
max_value = -1
for combination in product(*lists):
    result = sum(x ** 2 for x in combination) % M
    max_value = max(max_value, result)
print(max_value)
```


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)

```
3 1000
2 5 4
3 7 8 9
5 5 7 8 9 10
```

Your Output (stdout)

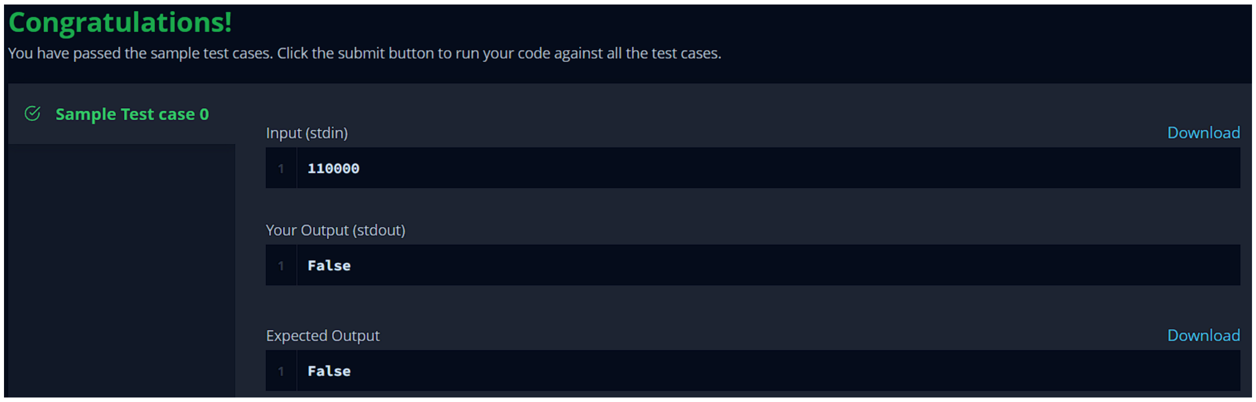
```
206
```

Expected Output

```
206
```

2. *Validating Postal Codes*

```
regex_integer_in_range = r'^[1-9][0-9]{5}$' # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r'(\d)(?=\d\1)'   # Do no
t 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
)
```



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)

```
110000
```

Your Output (stdout)

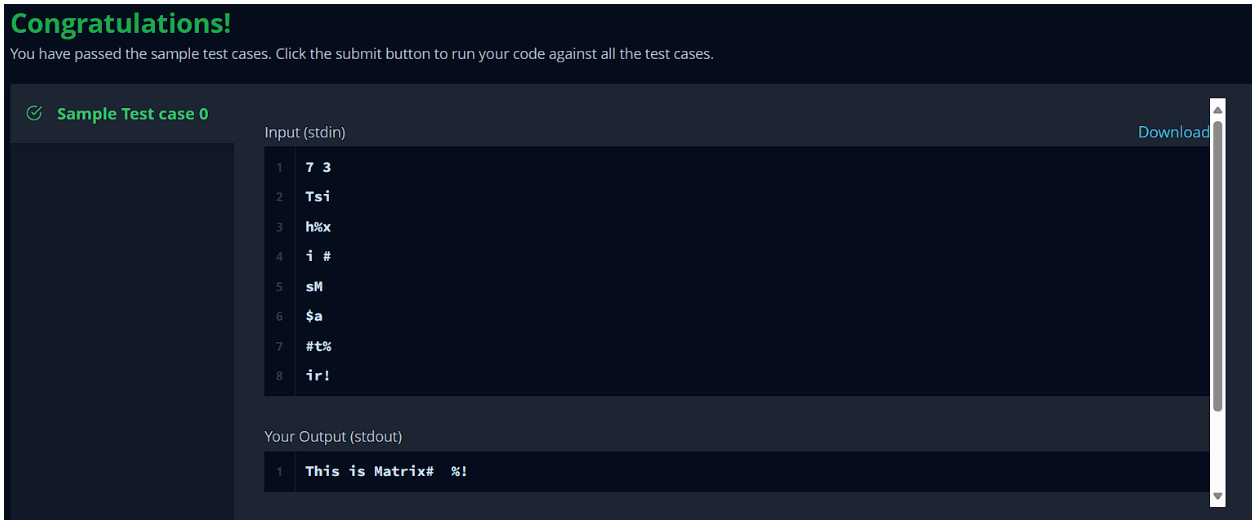
```
False
```

Expected Output

```
False
```

3. Matrix Script

```
import math
import os
import random
import re
import sys
first_multiple_input = input().rstrip().split()
n = int(first_multiple_input[0])
m = int(first_multiple_input[1])
matrix = []
for _ in range(n):
    matrix_item = input()
    matrix.append(matrix_item)
decoded = ''
for i in range(m):
    for j in range(n):
        decoded += matrix[j][i]
decoded = re.sub(r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-
9])', ' ', decoded)
print(decoded)
```



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

```
7 3
Tsi
h%x
i #
sM
$a
#t%
ir!
```

Your Output (stdout)

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