

**Assignment No 3**

**ZAHEER ABBAS**

**PHD MECHANICAL ENGINEERING**

**REG NO 433031**

**Maximize it – Hard**

```
import itertools
line = input()
K = int(line.split()[0])
M = int(line.split()[1])
N = []
for i in range(K):
    l = input().split()
    l = [ int(n) for n in l ]
    l = l[1:]
    N.append(l)
pro = list( itertools.product( *N ) )
maxi = 0
for item in pro:
    sum=0
    for num in item:
        sum += num**2
    modu = sum % M
    if (modu > maxi):
        maxi = modu
print (maxi)
```

**Problem**

You are given a function  $f(X) = X^2$ . You are also given  $K$  lists. The  $i^{th}$  list consists of  $N_i$  elements.

You have to pick one element from each list so that the value from the equation below is maximized:

$$S = (f(X_1) + f(X_2) + \dots + f(X_K)) \% M$$

$X_i$  denotes the element picked from the  $i^{th}$  list. Find the maximized value  $S_{max}$  obtained.

$\%$  denotes the modulo operator.

Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.

**Input Format**

The first line contains 2 space separated integers  $K$  and  $M$ .

The next  $K$  lines each contains an integer  $N_i$ , denoting the number of elements in the  $i^{th}$  list, followed by  $N_i$  space separated integers denoting the elements in the list.

**Test Cases**

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

**Hidden Test Case**

Unlock this testcase for 5 hacks.

**Unlock**

## Validating Postal codes - **Hard**

```
regex_integer_in_range = r"^([1-9][0-9]{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)
```

Browser tabs: Mail - Zahi, Google A..., (1) WhatsA..., AI Assignm..., Upload file..., Validating..., (808) How..., how to sav...

URL: <https://www.hackerrank.com/challenges/validating-postalcode/problem?isFullScreen=true>

HackerRank | Prepare > Python > Regex and Parsing > Validating Postal Codes | Exit Full Screen View

**Problem**

A valid postal code  $P$  have to fulfill both below requirements:

1.  $P$  must be a number in the range from 100000 to 999999 inclusive.
2.  $P$  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

**Submissions**

**Leaderboard**

**Congratulations**  
You solved this challenge. Would you like to challenge your friends? [Facebook](#) [Twitter](#) [LinkedIn](#) [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 110000 [Download](#)

Expected Output  
1 False [Download](#)

Windows taskbar: Type here to search, 12°C, 6:56 PM 1/5/2024

## Matrix Script - **Hard**

```
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 = []
t = []
for _ in range(n):
    matrix_item = [x for x in input()]
    matrix.append(matrix_item)

for i in range(m):
    for j in range(n):
        t.append(matrix[j][i])
s = ''.join(t)
path = re.compile(r'\b[ !@#$%^&]+\b', re.M)
k = re.sub(path, ' ', s)
print(k)
```

The screenshot shows the HackerRank interface for the 'Matrix Script' challenge. The problem description states that a matrix script is an  $N \times M$  grid of strings. The input is a 7x3 grid of characters. The decoded message is 'This\$#is% Matrix# %!'. The test cases are all passed, and the input for the test cases is shown in a text area.

Matrix Script

T	s	i
h	%	x
i		#
s	M	
\$	a	
#	t	%
i	r	!

Matrix Decoded

**This\$#is% Matrix# %!**

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 T s i
3 h % x
4 i #
5 s M
6 $ a
7 # t %
8 i r !
```

## Write a Function (Leap Year)-Medium

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

    if 1900 <= year <= 10**5:
        leap = False
        if year % 4 == 0 and year % 100 != 0:
            leap = True
        if year % 400 == 0:
            leap = True
    return leap

year = int(input())
```

Browser tabs: Mail - Z, Google, (1) What, AI Assign, Write a, Upload, Write a, (808) H, how to, +

URL: <https://www.hackerrank.com/challenges/write-a-function/problem?isFullScreen=true>

HackerRank | Prepare > Python > Introduction > Write a function | Exit Full Screen View

**Problem**

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:
  - 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

**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	1990
Your Output (stdout)	
1	False
Expected Output	Download
1	False

Windows taskbar: Type here to search, 8°C, 6:15 PM, 1/5/2024

## Sort Athlete -Medium

```
if __name__ == "__main__":
    n, m = input().strip().split(' ')
    n, m = [int(n), int(m)]
    arr = []
    for arr_i in range(n):
        arr_t = [int(arr_temp) for arr_temp in input().strip().split(' ')]
        arr.append(arr_t)
    k = int(input().strip())
    sorted_arr = sorted(arr, key = lambda x : x[k])
    for row in sorted_arr:
        print(' '.join(str(y) for y in row))
```

https://www.hackerrank.com/challenges/python-sort/problem?isFullScreen=true

HackerRank | Prepare > Python > Built-Ins > Athlete Sort

Exit Full Screen View

**Problem**

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  $K^{\text{th}}$  attribute and print the final resulting table. Follow the example given below for better understanding.

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

sort based on  $k=1$   
i.e (age)

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**

**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)

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

Your Output (stdout)

1	7	1 0
2	10	2 5
3	6	5 9

## Minion Game -Medium

```
def minion_game(string: str) -> None:
    kevin = stuart = 0
    length: int = len(string)
    for i, char in enumerate(string):
        points: int = length - i
        if char in {"A", "E", "I", "O", "U"}:
            kevin += points
        else:
            stuart += points
    if kevin == stuart:
        print("Draw")
```

```

else:
    print(*("Stuart", stuart) if stuart > kevin else ("Kevin", kevin))

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

```

The screenshot shows a web browser window displaying the HackerRank challenge page for 'The Minion Game'. The browser's address bar shows the URL: <https://www.hackerrank.com/challenges/the-minion-game/problem?isFullScreen=true>. The page header includes the HackerRank logo and navigation links: 'Prepare', 'Python', 'Strings', and 'The Minion Game'. On the left sidebar, there are links for 'Problem', 'Submissions', and 'Leaderboard'. The main content area on the left contains the problem description, game rules, scoring, and an example. On the right, a large green banner reads 'Congratulations' and 'You solved this challenge. Would you like to challenge your friends?'. Below this, a list of test cases shows 'Test case 0' through 'Test case 3' all marked as successful. The 'Input (stdin)' section shows the input 'BANANA'. The 'Compiler Message' section shows 'Success'. The 'Expected Output' section is empty. At the bottom of the page, there is a Windows taskbar with various application icons and a system tray showing the time as 6:32 PM on 1/5/2024.

## Time Delta -Medium

```

import math
import os
import random
import re
import sys
from datetime import datetime
def time_delta(t1, t2):
    format_ = '%a %d %b %Y %H:%M:%S %z'
    t1 = datetime.strptime(t1, format_)
    t2 = datetime.strptime(t2, format_)
    return str(int(abs((t1-t2).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()

```

The screenshot shows the HackerRank interface for the 'Time Delta' challenge. The page is titled 'HackerRank | Prepare > Python > Date and Time > Time Delta'. The problem description on the left explains that users post updates on social media, and the task is to calculate the absolute difference in seconds between two timestamps. The input format specifies that the first line is the number of test cases, and each test case consists of two lines representing timestamps. The constraints state that the input contains only valid timestamps.

The main content area displays a green 'Congratulations' banner with the text 'You solved this challenge. Would you like to challenge your friends?' and a 'Next Challenge' button. Below this, the test cases are listed:

- Test case 0: Success
- Test case 1: Success
- Test case 2: Success

The input (stdin) for the test cases is shown as follows:

```

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

```

The expected output is also shown, with a 'Download' button next to it.

## Merge the tools! -Medium

```

def merge_the_tools(string, k):
    l = len(string)//k
    for i in range(l):
        print(''.join(dict.fromkeys(string[i*k:(i*k)+k])))

if __name__ == '__main__':

```



```
string, k = input(), int(input())
merge_the_tools(string, k)
```

**HackerRank** | Prepare > Python > Strings > Merge the Tools!

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

**Problem**

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'

**Submissions**

**Leaderboard**

**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
```

12°C 6:48 PM 1/5/2024

## Classes : dealing with complex numbers - Medium

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

```

**HackerRank** | Prepare > Python > Classes > Classes: Dealing with Complex Numbers

**Problem**

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.

**Input Format**

One line of input: The real and imaginary part of a number separated by a space.

**Output Format**

For two complex numbers  $C$  and  $D$ , the output should be in the following sequence on separate lines:

- $C + D$
- $C - D$
- $C * D$
- $C / D$
- $mod(C)$
- $mod(D)$

For complex numbers with non-zero real( $A$ ) and complex part

**Congratulations**

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

**Test Cases**

- Test case 3 [View](#)
- Test case 4 [View](#)
- Test case 5 [View](#)
- Test case 6 [View](#)
- Test case 7 [View](#)
- Test case 8 [View](#)

**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

```

Windows taskbar: 11°C, 7:13 PM, 1/5/2024

## Company Logo - Medium

```
import math
import os
import random
import re
import sys
from collections import Counter
if __name__ == '__main__':
    s = input()
    s = sorted(s)
    f = Counter(list(s))
    for k, v in f.most_common(3):
        print(k, v)
```

The screenshot shows the HackerRank interface for the 'Company Logo' challenge. The left sidebar contains links for 'Problem', 'Submissions', and 'Leaderboard'. The main content area displays the problem description, input format, and constraints. On the right, a green 'Congratulations' banner indicates a successful submission. Below this, test cases 0 through 5 are listed, all marked as passed. The 'Compiler Message' shows 'Success'. The 'Input (stdin)' section shows the input string 'aabbccde'. The 'Expected Output' section shows the expected results for each character: 'b 3', 'a 2', and 'c 2'.

**Problem**

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 it's logo with the letters **G, O, E**.

**Input Format**

A single line of input containing the string *S*.

**Constraints**

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

**Test case 0** ✓

**Test case 1** ✓

**Test case 2** ✓

**Test case 3** ✓

**Test case 4** ✓

**Test case 5** ✓

**Compiler Message**  
Success

**Input (stdin)**  
1 aabbccde

**Expected Output**  
1 b 3  
2 a 2  
3 c 2

## Compress the string - Medium

```
from itertools import groupby
for a, b in groupby(input()):
    print("(%d, %d)" % (len(list(b)), int(a)), end=' ')
```

The screenshot shows the HackerRank interface for the 'Compress the String!' challenge. The problem description on the left explains that the task is to compress a string by replacing consecutive occurrences of a character 'c' with (X, c), where X is the number of consecutive occurrences. The input format is a single line of string S, and the output format is a single line of the modified string. Constraints state that all characters of S are integers between 0 and 9, and 1 ≤ |S| ≤ 10<sup>4</sup>. A sample input is provided.

On the right, a green 'Congratulations' banner indicates a successful submission. Below it, a list of 12 test cases is shown, all marked as passed. The 'Compiler Message' section displays 'Success'. The 'Input (stdin)' section shows the input '1 1222311', and the 'Expected Output' section shows the output '1 (1, 1) (3, 2) (1, 3) (2, 1)'. A 'Next Challenge' button is visible in the top right corner.

## Default Arguments - Medium

```
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())

raw_input = input

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())

```

Browser tabs: Mail - Zah... Google A... (1) WhatsA... AI Assignm... Upload file... Default Ar... (809) How... how to sa... | +

Address bar: <https://www.hackerrank.com/challenges/default-arguments/problem?isFullScreen=true>

HackerRank | Prepare > Python > Debugging > Default Arguments | Exit Full Screen View

**Problem**

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.

**Submissions**

**Leaderboard**

## 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)

```
1 3
2 odd 2
3 even 3
4 odd 5
```

Download

Your Output (stdout)

```
1 1
2 3
3 0
4 2
5 4
6 1
```

Windows taskbar: Type here to search | 11°C | 7:21 PM 1/5/2024

## Finding Angle - Medium

```
import math
def angle_MBC(AB, BC):
    theta_rad = math.atan2(AB, BC)
    theta_deg = round(theta_rad * (180 / math.pi))

    return theta_deg

AB = float(input())
BC = float(input())

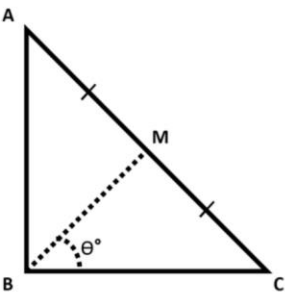
angle = angle_MBC(AB, BC)
print("{}\u00b0".format(angle))
```

Browser tabs: Mail - Zahi, Google Ac, (1) WhatsA, AI Assignm, Upload file, Find Angle, (809) How, how to sav

URL: <https://www.hackerrank.com/challenges/find-angle/problem?isFullScreen=true>

HackerRank | Prepare > Python > Math > Find Angle MBC | Exit Full Screen View

**Problem**



$ABC$  is a right triangle,  $90^\circ$  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  $\theta^\circ$ , as shown in the figure) in degrees.

**Submissions**

**Leaderboard**

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

**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	10
2	10

Expected Output [Download](#)

1	45°
---	-----

Windows taskbar: Type here to search | 11°C | 7:26 PM 1/5/2024



## Iterables and Iterators - Medium

```
from itertools import combinations, groupby

count, letters, select = int(input()), input().split(), int(input())
letters = sorted(letters)
combine = list(combinations(letters, select))
contain = len([c for c in combine if 'a' in c])
print(contain / len(combine))
```

The screenshot shows a web browser window displaying the HackerRank challenge page for 'Iterables and Iterators'. The page is in full-screen mode. On the left, the 'Problem' section describes the challenge: given a list of  $N$  lowercase English letters and an integer  $K$ , find the probability that at least one of the  $K$  selected indices contains the letter 'a'. The 'Input Format' section specifies that the input consists of three lines: the length  $N$ , the list of letters, and the integer  $K$ . The 'Submissions' and 'Leaderboard' sections are visible on the left sidebar.

The main content area shows a green 'Congratulations' banner with the text 'You solved this challenge. Would you like to challenge your friends?' and a 'Next Challenge' button. Below this, a list of test cases (Test case 5 to Test case 10) is shown, all with a green checkmark indicating success. To the right of the test cases, the 'Compiler Message' section shows 'Success'. Below that, the 'Input (stdin)' section displays the input for the test case: 4, a a c d, 2. The 'Expected Output' section shows the output: 0.833333333333. A 'Download' button is present next to the input and output sections.

The browser's address bar shows the URL: <https://www.hackerrank.com/challenges/iterables-and-iterators/problem?isFullScreen=true>. The Windows taskbar at the bottom shows the system clock as 7:28 PM on 1/5/2024, and the temperature as 11°C.



## No idea - Medium

```
if __name__ == "__main__":
    happy = 0
    n, m = map(int, input().strip().split(' '))
    elements_arr = list(map(int, input().strip().split(' ')))

    A = set(map(int, input().strip().split(' ')))
    B = set(map(int, input().strip().split(' ')))

    for i in elements_arr:
        if i in A:
            happy += 1
        elif i in B:
            happy -= 1
    print(happy)
```

The screenshot shows the HackerRank interface for the 'No idea' challenge. On the left, the problem description is visible, detailing the rules for calculating happiness based on set membership. The main area displays a green 'Congratulations' banner, indicating a successful solution. Below this, a list of six test cases is shown, all marked as passed. The 'Compiler Message' section shows 'Success'. The 'Input (stdin)' section displays the input data for the test cases, and the 'Expected Output' section shows the correct output for each case.

**Problem Description:**

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.

**Test Cases:**

- Test case 1: Passed
- Test case 2: Passed
- Test case 3: Passed
- Test case 4: Passed
- Test case 5: Passed
- Test case 6: Passed

**Compiler Message:** Success

**Input (stdin):**

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

**Expected Output:**

```
1
```