



ASSIGNMENT # 3

NAME : M. HAMZA ZAMAN
REG NO : 402581
DEGREE : MS (MECHANICAL ENGINEERING)
SUBMITTED TO : DR YASAR AYAZ
DATED : 05 JANUARY 2024

MEDIUM DIFFICULTY CHALLENGES

1. Write a Function

Code:

```
1  def is_leap(year):
2      leap = False
3      if year%400==0:
4          leap = True
5      elif year%100==0:
6          leap = False
7      elif year%4==0:
8          leap = True
9      else:
10         leap = False
11
12     # Write your logic here
13
14     return leap
15
16 > year = int(input()) ...
```

Output:

The screenshot shows a code execution environment with a sidebar on the left containing a list of test cases, each with a green checkmark and a lock icon. The main area on the right displays the results for 'Test case 0'. At the top, the 'Compiler Message' section shows 'Success'. Below that, the 'Input (stdin)' section shows a single line of input: '2000'. Further down, the 'Expected Output' section shows a single line of output: 'True'. At the bottom of the interface, there is a label for a 'Hidden Test Case'.

| Test Case | Compiler Message | Input (stdin) | Expected Output |
|------------------|------------------|---------------|-----------------|
| Test case 0 | Success | 2000 | True |
| Test case 1 | | | |
| Test case 2 | | | |
| Test case 3 | | | |
| Test case 4 | | | |
| Test case 5 | | | |
| Hidden Test Case | | | |

2. The Minion Game

Code:

```
1 def minion_game(string):
2     vowels = 'AEIOU'
3     keysc = 0
4     stusc = 0
5     for i in range(0, len(string)):
6         if string[i] in vowels:
7             keysc += len(string) - i
8         else:
9             stusc += len(string) - i
10
11     if keysc > stusc:
12         print('Kevin {}'.format(keysc))
13     elif stusc > keysc:
14         print('Stuart {}'.format(stusc))
15     else:
16         print('Draw')
17
18 if __name__ == '__main__':
19     s = input()
20     minion_game(s)
```

Output:

The screenshot shows a code execution interface with a sidebar on the left listing test cases from 0 to 6, all marked as successful with green checkmarks. The main area on the right displays the execution results for 'Test case 0'. It includes a 'Compiler Message' section showing 'Success', an 'Input (stdin)' section showing 'BANANA', and an 'Expected Output' section showing 'Stuart 12'. There are 'Download' links next to the input and expected output sections.

| Test Case | Status | Input (stdin) | Expected Output |
|-------------|---------|---------------|-----------------|
| Test case 0 | Success | BANANA | Stuart 12 |
| Test case 1 | Success | | |
| Test case 2 | Success | | |
| Test case 3 | Success | | |
| Test case 4 | Success | | |
| Test case 5 | Success | | |
| Test case 6 | Success | | |

3. Merge the Tools!

Code:

```
1 def merge_the_tools(string, k):
2     for i in range(0, len(string), k):
3         unique_list = []
4         str_list = list(string[i:i+k])
5         for c in str_list:
6             if c not in unique_list:
7                 unique_list.append(c)
8         print("".join(unique_list))
9
10 if __name__ == '__main__':
11     string, k = input(), int(input())
12     merge_the_tools(string, k)
```

Output:

The screenshot shows a code execution interface with a sidebar on the left containing a list of test cases, all marked as successful with green checkmarks. The main area on the right displays the compiler message 'Success', the input (stdin) as 'AABCAAADA' on line 1 and '3' on line 2, and the expected output as 'AB' on line 1, 'CA' on line 2, and 'AD' on line 3. There are 'Download' links next to the input and output sections.

| 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):

| | |
|---|-----------|
| 1 | AABCAAADA |
| 2 | 3 |

Expected Output:

| | |
|---|----|
| 1 | AB |
| 2 | CA |
| 3 | AD |

4. Time Delta

Code:

```
1  #!/bin/python3
2  import math
3  import os
4  import random
5  import re
6  import sys
7  # Complete the time_delta function below.
8  from datetime import datetime
9  def time_delta(t1, t2):
10     time_format = '%a %d %b %Y %H:%M:%S %z'
11     t1 = datetime.strptime(t1, time_format)
12     t2 = datetime.strptime(t2, time_format)
13     return str(int(abs((t1-t2).total_seconds())))
14 if __name__ == '__main__':
15     fptr = open(os.environ['OUTPUT_PATH'], 'w')
16     t = int(input())
17     for t_itr in range(t):
18         t1 = input()
19         t2 = input()
20         delta = time_delta(t1, t2)
21         fptr.write(delta + '\n')
22     fptr.close()
23
```

Output:

The screenshot shows a code execution environment with a dark theme. On the left, there are three test cases, all marked as successful with green checkmarks: 'Test case 0', 'Test case 1', and 'Test case 2'. The main area is divided into sections. The 'Compiler Message' section shows 'Success'. The 'Input (stdin)' section shows a list of 5 inputs: '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', and 'Fri 01 May 2015 13:54:36 -0000'. The 'Expected Output' section shows a single output: '25200'. There are 'Download' links next to the input and expected output sections.

| Test Case | Input (stdin) | Expected Output |
|-------------|---------------|-----------------|
| Test case 0 | 2 | 25200 |

5. Find Angle MBC

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import math
3 ab=int(input())
4 bc=int(input())
5 ca=math.hypot(ab, bc)
6 mc=ca/2
7 bca=math.asin(1*ab/ca)
8 bm=math.sqrt((bc**2+mc**2)-(2*bc*mc*math.cos(bca)))
9 mbc=math.asin(math.sin(bca)*mc/bm)
10 print(int(round(math.degrees(mbc),0)), '\u00B0', sep='')
```

Output:

✔ 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° |
|---|-----|

6. No Idea!

Code:

```
1  # Enter your code here. Read input from STDIN. Print output to STDOUT
2  def happiness(input_array, A, B):
3      happiness_score = 0
4      for i in input_array:
5          if i in A:
6              happiness_score += 1
7          if i in B:
8              happiness_score -= 1
9      return happiness_score
10
11
12  def main():
13      n, m = input().split()
14      input_array = list(map(int, input().split()))
15      A = set(map(int, input().split()))
16      B = set(map(int, input().split()))
17      print(happiness(input_array, A, B))
18
19
20  if __name__ == '__main__':
21      main()
```

Output:

The screenshot shows a coding competition interface with a dark theme. On the left, a sidebar lists seven test cases, all marked as successful with green checkmarks. The main area is divided into three sections: 'Compiler Message' showing 'Success', 'Input (stdin)' showing four lines of input, and 'Expected Output' showing one line of output. Each section has a 'Download' link on the right.

| 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 |
| 2 | 1 5 3 |
| 3 | 3 1 |
| 4 | 5 7 |

Expected Output [Download](#)

| | |
|---|---|
| 1 | 1 |
|---|---|

7. Word Order

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 from collections import Counter
3 N = int(input())
4 LIST = []
5 for i in range(N):
6     LIST.append(input().strip())
7 COUNT = Counter(LIST)
8 print(len(COUNT))
9 print(*COUNT.values())
```

Output:

The screenshot displays a coding platform interface with a sidebar on the left and a main content area on the right. The sidebar lists seven test cases, all marked as successful with green checkmarks. The main content area shows the input and expected output for the first test case.

Test case 0 (Success)

Input (stdin)

| | |
|---|---------|
| 1 | 4 |
| 2 | bcdef |
| 3 | abcdefg |
| 4 | bcde |
| 5 | bcdef |

Expected Output

| | |
|---|-------|
| 1 | 3 |
| 2 | 2 1 1 |

8. Compress the String!

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import itertools
3
4 print(*[ ( len(list(g)), int(k) ) for k, g in itertools.groupby(input()) ])
```

Output:

✓ 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 1222311

Expected Output [Download](#)

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

9. Company Logo

Code:

```
1 from collections import Counter
2 S = input()
3 S = sorted(S)
4 FREQUENCY = Counter(list(S))
5 ✓ for k, v in FREQUENCY.most_common(3):
6     print(k, v)
```

Output:

✓ 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 aabbccde

Expected Output [Download](#)

1 b 3
2 a 2
3 c 2

10. Piling Up!

Code:

```
1  # Enter your code here. Read input from STDIN. Print output to STDOUT
2  ANS = []
3  T = int(input())
4  for _ in range(T):
5      n = int(input())
6      sl = list(map(int, input().split()))
7      for _ in range(n-1):
8          if sl[0] >= sl[len(sl)-1]:
9              a = sl[0]
10             sl.pop(0)
11         elif sl[0] < sl[len(sl)-1]:
12             a = sl[len(sl)-1]
13             sl.pop()
14         else:
15             len(obj: Sized, /) -> int
16             Return the number of items in a container.
17             View Problem (Alt+F8) No quick fixes available
18         if ((sl[0] > a) or (sl[len(sl)-1] > a)):
19             ANS.append("No")
20             break
21 print("\n".join(ANS))
```

Output:

| | |
|-----------------|-------------|
| ✓ Test case 0 | Success |
| ✓ Test case 1 | |
| ✓ Test case 2 | |
| ✓ Test case 3 | |
| ✓ Test case 4 | |
| Input (stdin) | |
| 1 | 2 |
| 2 | 6 |
| 3 | 4 3 2 1 3 4 |
| 4 | 3 |
| 5 | 1 3 2 |
| Expected Output | |
| 1 | Yes |
| 2 | No |

11. Triangle Quest 2

Code:

```
1  
2  
3  
4  
5 ∨ for i in range(1,int(input())+1):  
6   print(pow(((pow(10,i)-1)//9), 2))
```

Output:

✓ 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 |

12. Iterables and Iterators

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 from itertools import combinations
3 N = int(input())
4 LETTERS = list(input().split(" "))
5 K = int(input())
6 TUPLES = list(combinations(LETTERS, K))
7 CONTAINS = [word for word in TUPLES if "a" in word]
8 print(len(CONTAINS)/len(TUPLES))
```

Output:

✔ 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 | 4 |
| 2 | a a c d |
| 3 | 2 |

Expected Output [Download](#)

| | |
|---|----------------|
| 1 | 0.833333333333 |
|---|----------------|

13. Triangle Quest

Code:

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

Output:

✓ 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

14. Classes: Dealing with Complex Numbers

Code:

```
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):
        r = (self.real*no.real)-(self.imaginary*no.imaginary)
        i = (self.real*no.imaginary+no.real*self.imaginary)
        return Complex(r, i)
    def __truediv__(self, no):
        conjugate = Complex(no.real, (-no.imaginary))
        num = self*conjugate
        denom = no*conjugate
        try:
            return Complex((num.real/denom.real), (num.imaginary/denom.real))
        except Exception as e:
            print(e)
    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
34
35
36
37
38
39 if __name__ == '__main__':
40     c = map(float, input().split())
41     d = map(float, input().split())
42     x = Complex(*c)
43     y = Complex(*d)
44     print(*map(str, [x+y, x-y, x*y, x/y, x.mod(), y.mod()]), sep='\n')
```

Line: 31 Col

Output:

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

✓ Test case 6

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 |
| 5 | 2.24+0.00i |
| 6 | 7.81+0.00i |

✓ 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 |

15. Athlete Sort

Code:

```
1  #!/bin/python3
2  import math
3  import os
4  import random
5  import re
6  import sys
7  N, M = map(int, input().split())
8  rows = [input() for _ in range(N)]
9  K = int(input())
10 for row in sorted(rows, key=lambda row: int(row.split()[K])):
11     print(row)
```

Output:

✓ Test case 0

✓ Test case 1

Compiler Message

Success

Input (stdin)

Download

| | |
|---|---------|
| 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 |

Expected Output

Download

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

16.ginortS

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 print(*sorted(input(), key=lambda c: (c.isdigit() - c.islower(), c in '02468', c)), sep='')
```

Output:

✓ 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 Sorting1234

Expected Output

Download

1 ginortS1324

17. Validating Email Addresses With a Filter

Code:

```
1 def fun(email):
2     try:
3         username, url = email.split('@')
4         website, extension = url.split('.')
5     except ValueError:
6         return False
7     if username.replace('-', '').replace('_', '').isalnum() is False:
8         return False
9     elif website.isalnum() is False:
10        return False
11    elif len(extension) > 3:
12        return False
13    else:
14        return True
15
16 def filter_mail(emails):
17     return list(filter(fun, emails))
18
19 if __name__ == '__main__':
20     n = int(input())
21     emails = []
22     for _ in range(n):
23         emails.append(input())
24
25 filtered_emails = filter_mail(emails)
26 filtered_emails.sort()
27 print(filtered_emails)
```

Output:

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 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']

18.Reduce Function

Code:

```
1  ✓ from fractions import Fraction
2  from functools import reduce
3  def product(fracs):
4      t = Fraction(reduce(lambda x, y: x * y, fracs))
5      return t.numerator, t.denominator
6
7  ✓ if __name__ == '__main__':
8      fracs = []
9      for _ in range(int(input())):
10         fracs.append(Fraction(*map(int, input().split())))
11     result = product(fracs)
12     print(*result)
```

Output:

✓ 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  21
2  684025282 932952183
3  349232934 278093065
4  778706161 742081687
5  374870211 874099626
6  849763633 211127281
7  566205501 508794028
8  814324820 443967409
9  402053385 120666811
```

19.Regex Substitution

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 for _ in range(int(input())):
4     print(re.sub(r'(?<= )(&&|\||\|)(?= )', lambda x: 'and' if x.group() == '&&' else 'or',
5               input()))
```

Output:

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 11
2 a = 1;
3 b = input();
4
5 if a + b > 0 && a - b < 0:
6     start()
7 elif a*b > 10 || a/b < 1:
8     stop()
9 print set(list(a)) | set(list(b))
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Expected Output

```
1 a = 1;
2 b = input();
3
4 if a + b > 0 and a - b < 0:
5     start()
6 elif a*b > 10 or a/b < 1:
7     stop()
8 print set(list(a)) | set(list(b))
9 #Note do not change &&& or ||| or & or |
10 #Only change those '&&' which have space on both sides.
11 #Only change those '||' which have space on both sides.
```

20. Validating Credit Card Numbers

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import re
3 n = int(input())
4 for t in range(n):
5     credit = input().strip()
6     credit_removed_hiphen = credit.replace('-', '')
7     valid = True
8     length_16 = bool(re.match(r'^[4-6]\d{15}$', credit))
9     length_19 = bool(re.match(r'^[4-6]\d{3}-\d{4}-\d{4}-\d{4}$', credit))
10    consecutive = bool(re.findall(r'(?=(\d)\1\1\1)', credit_removed_hiphen))
11    if length_16 == True or length_19 == True:
12        if consecutive == True:
13            valid = False
14    else:
15        valid = False
16    if valid == True:
17        print('Valid')
18    else:
19        print('Invalid')
```

Output:

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

Compiler Message

Success

Input (stdin)

| | |
|---|---------------------------|
| 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 |

Download

✓ Test case 3

✓ Test case 4

✓ Test case 5

Expected Output

| | |
|---|---------|
| 1 | Valid |
| 2 | Valid |
| 3 | Invalid |
| 4 | Valid |
| 5 | Invalid |
| 6 | Invalid |

Download

21. Words Score

Code:

```
1 def is_vowel(letter):
2     return letter in ['a', 'e', 'i', 'o', 'u', 'y']
3 def is_vowel(letter):
4     return letter in ['a', 'e', 'i', 'o', 'u', 'y']
5 def score_words(words):
6     score = 0
7     for word in words:
8         num_vowels = 0
9         for letter in word:
10            if is_vowel(letter):
11                num_vowels += 1
12            if num_vowels % 2 == 0:
13                score += 2
14            else:
15                score += 1
16    return score
17
18 n = int(input())
19 words = input().split()
20 print(score_words(words))
```

Output:

The screenshot shows a code execution environment with a sidebar on the left containing a list of test cases, all marked as successful with green checkmarks. The main area on the right displays the execution results for 'Test case 0'. It includes a 'Compiler Message' section showing 'Success', an 'Input (stdin)' section showing the input '2' followed by 'hacker book' on the next line, and an 'Expected Output' section showing the output '4'. Each of these sections has a 'Download' link to its right.

| 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 |

Test case 0

Compiler Message: Success

Input (stdin):

```
1 2
2 hacker book
```

Expected Output:

```
1 4
```

22. Default Arguments

Code:

```
1 class EvenStream(object):
2     def __init__(self):
3         self.current = 0
4
5     def get_next(self):
6         to_return = self.current
7         self.current += 2
8         return to_return
9
10 class OddStream(object):
11     def __init__(self):
12         self.current = 1
13
14     def get_next(self):
15         to_return = self.current
16         self.current += 2
17         return to_return
18
19 def print_from_stream(n, stream=EvenStream()):
20     stream.__init__()
21     for _ in range(n):
22         print(stream.get_next())
23
24 queries = int(input())
25 for _ in range(queries):
26     stream_name, n = input().split()
27     n = int(n)
28     if stream_name == "even":
29         print_from_stream(n)
30     else:
31         print_from_stream(n, OddStream())
32
33
```

Output:

✓ 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 | 3 |
| 2 | odd 2 |
| 3 | even 3 |
| 4 | odd 5 |

Expected Output[Download](#)

| | |
|---|---|
| 1 | 1 |
| 2 | 3 |

HARD DIFFICULTY CHALLENGES

1. Maximize It!

Code:

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import itertools
3 NUMBER_OF_LISTS, MODULUS = map(int, input().split())
4 LISTS_OF_LISTS = []
5 for i in range(0, NUMBER_OF_LISTS):
6     new_list = list(map(int, input().split()))
7     del new_list[0]
8     LISTS_OF_LISTS.append(new_list)
9 def squared(element):
10     return element**2
11 COMBS = list(itertools.product(*LISTS_OF_LISTS))
12 RESULTS = []
13 for i in COMBS:
14     result1 = sum(map(squared, [a for a in i]))
15     result2 = result1 % MODULUS
16     RESULTS.append(result2)
17 print(max(RESULTS))
```

Output:

The screenshot shows a coding challenge interface. On the left, there is a list of test cases: Test case 0, Test case 1, Test case 2, Test case 3, Test case 4, Test case 5, and Test case 6. Each test case is preceded by a green checkmark and followed by a lock icon. The main area on the right is titled 'Compiler Message' and shows a 'Success' message. Below this, there is a section titled 'Hidden Test Case' with the text 'Unlock this testcase for 5 hackos.' and a button labeled 'Unlock'.

2. Validating Postal Codes

Code:

```
1 regex_integer_in_range = r"[1-9][\d]{5}$" # Do not delete 'r'.
2 regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)" # Do not delete 'r'.
3
4 import re
5 P = input()
6
7 print(bool(re.match(regex_integer_in_range, P))
8       and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

Output:

✓ 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 110000

Expected Output[Download](#)

1 False

3. Matrix Script

Code:

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8
9  n, m = map(int, input().split())
10 character_ar = [' '] * (n*m)
11 for i in range(n):
12     line = input()
13     for j in range(m):
14         character_ar[i+(j*n)] = line[j]
15 decoded_str = ''.join(character_ar)
16 final_decoded_str = re.sub(r'(?<=[A-Za-z0-9])([ !@#$%&]+)(?=[A-Za-z0-9])', ' ', decoded_str)
17 print(final_decoded_str)
18
```

Output:

| | |
|---------------|---|
| ✓ Test case 0 | Compiler Message |
| ✓ Test case 1 | Success |
| ✓ Test case 2 | Input (stdin) Download |
| ✓ Test case 3 | <pre>1 7 3 2 Tsi 3 h%x 4 i # 5 sM 6 \$a 7 #t% 8 ir!</pre> |
| ✓ Test case 5 | Expected Output Download |
| ✓ Test case 6 | <pre>1 This is Matrix# %!</pre> |

ALL SOLVED CHALLENGES

| | | |
|--|---|----------|
| Write a function <small>Medium, Python (Basic), Max Score: 10, Success Rate: 90.33%</small> | ★ | Solved ✓ |
| The Minion Game <small>Medium, Python (Basic), Max Score: 40, Success Rate: 86.80%</small> | ★ | Solved ✓ |
| Merge the Tools! <small>Medium, Problem Solving (Basic), Max Score: 40, Success Rate: 93.76%</small> | ★ | Solved ✓ |
| Time Delta <small>Medium, Python (Basic), Max Score: 30, Success Rate: 91.36%</small> | ★ | Solved ✓ |
| Find Angle MBC <small>Medium, Python (Basic), Max Score: 10, Success Rate: 89.16%</small> | ★ | Solved ✓ |
| No Idea! <small>Medium, Python (Basic), Max Score: 50, Success Rate: 88.03%</small> | ★ | Solved ✓ |
| Word Order <small>Medium, Python (Basic), Max Score: 50, Success Rate: 90.24%</small> | ★ | Solved ✓ |
| Compress the String! <small>Medium, Python (Basic), Max Score: 20, Success Rate: 97.15%</small> | ★ | Solved ✓ |
| Company Logo <small>Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.84%</small> | ★ | Solved ✓ |

Piling Up!

Medium, Python (Basic), Max Score: 50, Success Rate: 90.64%



Solved

Triangle Quest 2

Medium, Python (Basic), Max Score: 20, Success Rate: 95.38%



Solved

Iterables and Iterators

Medium, Python (Basic), Max Score: 40, Success Rate: 96.60%



Solved

Triangle Quest

Medium, Python (Basic), Max Score: 20, Success Rate: 93.84%



Solved

Classes: Dealing with Complex Numbers

Medium, Python (Basic), Max Score: 20, Success Rate: 90.92%



Solved

Athlete Sort

Medium, Python (Basic), Max Score: 30, Success Rate: 95.53%



Solved

ginortS

Medium, Python (Basic), Max Score: 40, Success Rate: 97.63%



Solved

Validating Email Addresses With a Filter

Medium, Python (Basic), Max Score: 20, Success Rate: 90.82%



Solved

Reduce Function

Medium, Max Score: 30, Success Rate: 98.38%



Solved

Regex Substitution

Medium, Python (Basic), Max Score: 20, Success Rate: 94.12%



Solved

Validating Credit Card Numbers

Medium, Python (Basic), Max Score: 40, Success Rate: 95.47%



Solved

Words Score

Medium, Max Score: 10, Success Rate: 94.94%



Solved

Default Arguments

Medium, Python (Intermediate), Max Score: 30, Success Rate: 78.83%



Solved

Maximize It!

Hard, Problem Solving (Basic), Max Score: 50, Success Rate: 81.27%



Solved

Validating Postal Codes

Hard, Max Score: 80, Success Rate: 87.40%



Solved

Matrix Script

Hard, Problem Solving (Advanced), Max Score: 100, Success Rate: 89.98%



Solved