NUST School of Mechanical and Manufacturing Engineering (SMME)

Assignment No. 3

Report on

Medium and Hard Challenges (Python) from HackerRank

Course: CSE-860 Artificial Intelligence

Submitted by: Mahak Khalid

Regn No: 450160

Medium Challenges

Challenge No: 1 Introduction (Write a function)

```
def is_leap(year):
    leap = False
    # Write your logic here
    if (year % 4 == 0 and
year % 100 != 0) or
(year % 400 == 0):
        leap = True
    else:
        leap = False

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

Sample Test case 0

Input (stdin)

```
1990
```

Your Output (stdout)

```
False
```

Expected Output

```
False
```

Challenge No: 2 Merge the Tools

```
def merge_the_tools(string, k):

    # your code goes here
    No_of_substrings = len(string) // k
    for i in range(No_of_substrings):
        substring = string[i * k : (i + 1) * k]
        unique_char = []
        for char in substring:
            if char not in unique_char:
                unique_char.append(char)
        print("".join(unique_char))
if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

Sample Test case 0

Input (stdin)

```
AABCAAADA
3
```

Your Output (stdout)

```
AB
CA
AD
```

Expected Output

```
AB
CA
AD
```

Challenge No:3 Date and Time (Time Delta)

```
import math
import os
import random
import re
import sys
```

```python
from datetime import datetime
def time_delta(t1, t2):
    format_str = "%a %d %b %Y %H:%M:%S %z"
    dt1 = datetime.strptime(t1, format_str)
    dt2 = datetime.strptime(t2, format_str)
    delta_seconds = int(abs((dt1 - dt2).total_seconds()))
    return str(delta_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()
```

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
```

<u>Challenge No:3</u> Math (Find Angle MBC)

```python
import math


def angle_MBC(AB, BC):
    angle = math.degrees(math.atan2(AB, BC))
    return round(angle)

if __name__ == '__main__':
    AB = float(input())
    BC = float(input())
    print(f"{angle_MBC(AB, BC)}\u00b0")
```

Input (stdin)

```
10
10
```

Your Output (stdout)

```
45°
```

Expected Output

```
45°
```

Challenge No:4 Sets (No Idea)

```
if __name__ == '__main__':
    n, m = map(int, input().split())
    array = list(map(int, input().split()))
    set_A = set(map(int, input().split()))
    set_B = set(map(int, input().split()))
    happiness = 0
    for i in array:
        if i in set_A:
            happiness += 1
        elif i in set_B:
            happiness -= 1
    print(happiness)
```

Sample Test case 0

Input (stdin)

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

Your Output (stdout)

```
1
```

Expected Output

```
1
```

Challenge No:5 Collections (Word Order)

```
from collections import OrderedDict

if __name__ == '__main__':
    m = int(input(""))
    word_counts = OrderedDict()
    for _ in range(m):
        word = input().strip()
        word_counts[word] =word_counts.get(word,0)+1
    print(len(word_counts))
    print(*word_counts.values())
```

```
4
bcdef
abcdefg
bcde
bcdef
```

Your Output (stdout)

```
3
2 1 1
```

Expected Output

```
3
2 1 1
```

Challenge No:6 Itertools (Compress the String)

```
from itertools import groupby

def compress_string(s):
    cmpd_string = []
    for char, group in groupby(s):
        count = len(list(group))
        cmpd_string.append(f"({count},{char}) ")
    return ''.join(compressed_string)
if __name__ == '__main__':
    s = input()
    modified_string = compress_string(s)
    print(modified_string)
```

Input (stdin) Download

```
1222311
```

Your Output (stdout)

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

Expected Output Download

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

Challenge No:7 Collections (Company LOGO)

```
import math
import os
import random
import re
import sys
from collections import Counter

if __name__ == '__main__':

    s = input()
    char_count = Counter(s)

    sorted_char = sorted(char_count.items(), key=lambda x: (-
x[1], x[0]))

    for char, count in sorted_char[:3]:
        print(f"{char} {count}")
```

Input (stdin)

```
aabbbccde
```

Your Output (stdout)

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

Expected Output

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

Challenge No:8 Collections (Piling Up!)

```
if __name__ == '__main__':
    T = int(input())
    for _ in range(T):
        n = int(input())
        cube = list(map(int, input().split()))
        left = 0
        right = n - 1
        curr_slen = float('inf')
        while left <= right:
            l_cube = cube[left]
            r_cube = cube[right]
            if l_cube >= r_cube and l_cube <= curr_slen:
                curr_slen = l_cube
                left += 1
            elif r_cube >= l_cube and r_cube <= curr_slen:
                curr_slen = r_cube
                right -= 1
            else:
                print("No")
                break
```

```
2
6
4 3 2 1 3 4
3
1 3 2
```

Your Output (stdout)

```
Yes
No
```

Expected Output

```
Yes
No
```

```
        else:
            print("Yes")
```

Challenge No:9 Math (Triangle Quest 2)

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

Sample Test case 0

Compiler Message

Success

Input (stdin)

```
5
```

Your Output (stdout)

```
1
121
12321
1234321
123454321
```

Expected Output

```
1
121
12321
1234321
123454321
```

Challenge No:10 Math (Triangle Quest)

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

Sample Test case 0

Compiler Message

Success

Input (stdin)

```
5
```

Your Output (stdout)

```
1
22
333
4444
```

Expected Output

```
1
22
333
4444
```

Challenge No:11 Classes (Classes: Dealing with Complex numbers)

```
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):
        real_part = self.real * no.real - self.imaginary * no.imaginary
```

```python
        imaginary_part = self.real * no.imaginary + self.imaginary * no.real
        return Complex(real_part, imaginary_part)

    def __truediv__(self, no):
        conjugate = Complex(no.real, -no.imaginary)
        numerator = self * conjugate
        denominator = no * conjugate
        return Complex(numerator.real / denominator.real, numerator.imaginary
 / denominator.real)

    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')
```

Sample Test case 0

Sample Test case 1

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
```

Expected Output

```
7.00+7.00i
-3.00-5.00i
4.00+17.00i
0.26-0.11i
2.24+0.00i
7.81+0.00i
```

Sample Test case 0

Sample Test case 1

Input (stdin)

```
5.9 6
9 10
```

Your Output (stdout)

```
14.90+16.00i
-3.10-4.00i
-6.90+113.00i
0.62-0.03i
8.41+0.00i
13.45+0.00i
```

Expected Output

```
14.90+16.00i
-3.10-4.00i
-6.90+113.00i
0.62-0.03i
8.41+0.00i
13.45+0.00i
```

Challenge No:12 built-ins(ginortS)

```
def custom_sort(c):
    if c.islower():
        return (0, c)
    elif c.isupper():
        return (1, c)
    elif c.isdigit():
        if int(c) % 2 == 1:
            return (2, int(c))
        else:
            return (3, int(c))

if __name__ == '__main__':
    s = input()
    result = ''.join(sorted(s, key=custom_sort))
    print(result)
```

Sample Test case 0

Input (stdin)

```
Sorting1234
```

Your Output (stdout)

```
ginortS1324
```

Expected Output

```
ginortS1324
```

Challenge No:13 Pyhton Functionals (Validating Email Addresses with a Filter)

```
import re
def fun(s):
    pattern = r'^[a-zA-Z0-9_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'
    return re.match(pattern, s) is not None
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)
```

Sample Test case 0

Sample Test case 1

```
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 Dow

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

Sample Test case 0

Sample Test case 1

Input (stdin)

```
2
harsh@gmail
iota_98@hackerrank.com
```

Your Output (stdout)

```
['iota_98@hackerrank.com']
```

Expected Output

```
['iota_98@hackerrank.com']
```

Challenge No:14 Python Functionals (Reduced Functions)

```python
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)
```

Sample Test case 0

Input (stdin)

```
3
1 2
3 4
10 6
```

Your Output (stdout)

```
5 8
```

Expected Output

```
5 8
```

Challenge No:15 Regex and Parsing (Regex substitution)

```python
import re

if __name__ == '__main__':
    N = int(input())
    text = [input() for _ in range(N)]

    modified_text = [re.sub(r'\s&&\s', ' and ', re.sub(r'\s\|\|\s',
 ' or ', line)) for line in text]

    for line in modified_text:
        print(line)
```

Sample Test case 0

Input (stdin)

```
11
a = 1;
b = input();

if a + b > 0 && a - b < 0:
    start()
elif a*b > 10 || 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.
```

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.
```

Expected Output

```
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.
```

Challenge No:16 Strings (The Minion Game)

```
def minion_game(string):

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

Sample Test case 0

Input (stdin)

```
BANANA
```

Your Output (stdout)

```
Stuart 12
```

Expected Output

```
Stuart 12
```

Challenge No:17 itertools (iterables and iterators)

```
from itertools import combinations

def probability_of_a(N, letters, K):
    total_combinations = list(combinations(letters, K))
    a_not_present_combinations = [comb for comb in total_combinatio
ns if 'a' not in comb]

    probability_not_a = len(a_not_present_combinations) / len(total
_combinations)
    probability_a = 1 - probability_not_a

    return round(probability_a, 3)
if __name__ == '__main__':
    N = int(input())
    letters = input().split()
    K = int(input())

    result = probability_of_a(N, letters, K)
    print(result)
```

Sample Test case 0

Input (stdin)

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

Your Output (stdout)

```
0.833
```

Expected Output

```
0.833333333333
```

<u>Challenge No:18</u> Regex and Parsing (Regex Substitution)

```
import re
def modify_symbols(text):
    pattern_and = re.compile(r'\s&&\s')
    pattern_or = re.compile(r'\s\|\|\s')
    modified_text = [pattern_and.sub(' and ', line) for line in tex
t]
    modified_text = [pattern_or.sub(' or ', line) for line in modif
ied_text]
    return modified_text
if __name__ == '__main__':
    N = int(input())
    text = [input() for _ in range(N)]
    modified_text = modify_symbols(text)
    for line in modified_text:
        print(line)
```

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.
```

<u>Challenge No:19</u> Built-ins(Athlete Sort)

```
#!/bin/python3

import math
import os
import random
import re
import sys
if __name__ == '__main__':
```

Test case 0

Test case 1

Compiler Message

Success

```
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())
arr.sort(key=lambda x: x[k])
for row in arr:
    print(*row)
```

Input (stdin)

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

Expected Output

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

<u>Challenge No:20</u> Regex and Parsing (Validating Credit Card Numbers)

```
import re

def is_valid_credit_card(card_number):

    pattern = r'^[456]\d{3}(-?\d{4}){3}$'

    if re.match(pattern, card_number):

        cleaned_number = card_number.replace('-', '')
        if re.search(r'(\d)\1{3,}', cleaned_number):
            return 'Invalid'
        else:
            return 'Valid'
    else:
        return 'Invalid'

if __name__ == '__main__':
    n = int(input().strip())

    for _ in range(n):
        card_number = input().strip()
        result = is_valid_credit_card(card_number)
        print(result)
```

Sample Test case 0

Input (stdin)

```
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
Invalid
```

Expected Output

```
Valid
Valid
Invalid
Valid
Invalid
Invalid
```

<u>Challenge No:21</u> Debugging (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))
```

Sample Test case 0

Sample Test case 1

Input (stdin)

```
2
hacker book
```

Your Output (stdout)

```
4
```

Expected Output

```
4
```

<u>Challenge No:22</u> Debugging (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
```

```python
def print_from_stream(n, stream=EvenStream()):
    if stream is None:
        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())
```

Sample Test case 0

Input (stdin)

```
3
odd 2
even 3
odd 5
```

Your Output (stdout)

```
1
3
0
2
4
1
3
5
7
9
```

Expected Output

```
1
3
0
2
4
1
3
5
7
9
```

Hard Challenges

Challenge No:23 Itertools (Maximize it!)

```python
from itertools import product

def maximize_S(K, M, lists):
    result = 0
    for combination in product(*lists):
        s = sum(x ** 2 for x in combination) % M
        result = max(result, s)
    return result
if __name__ == '__main__':
    K, M = map(int, input().split())
    lists = [list(map(int, input().split()[1:])) for _ in range(K)]
    result = maximize_S(K, M, lists)
    print(result)
```

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
```

Challenge No:24 Regex and Parsing (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 not
 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)
```

Sample Test case 0

Input (stdin)

```
110000
```

Your Output (stdout)

```
False
```

Expected Output

```
False
```

Challenge No:25 Regex and Parsing (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_script = ''
for j in range(m):
```

Sample Test case 0

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

Your Output (stdout)

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

Expected Output

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

```
    for i in range(n):
        decoded_script += matrix[i][j]

decoded_script = re.sub(r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-
9])', ' ', decoded_script)

print(decoded_script)
```