**Name: Muhammad Usman Zaib**

**Registration Number: 402104**

**Assignment No: 3**

**Total Number of Medium Task Solved = 22**

**Total Number of Hard Task Solved = 3**

## MEDIUM TASKS:

**1)Write a Function**

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

    return leap

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

**2)The Minion Game**

```python
def minion_game(string):
    vowels = "AEIOU"
    length = len(string)
    kevin_score = 0
    stuart_score = 0

    for i in range(length):
        if string[i] in vowels:
            kevin_score += length - i
        else:
            stuart_score += length - i

    # Determine the winner
    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)
```

3) **Merge the Tool**

```python
def merge_the_tools(string, k):
    # your code goes here
    temp = []
    len_temp = 0
    for item in string:
        len_temp += 1
        if item not in temp:
            temp.append(item)
        if len_temp == k:
            print (''.join(temp))
            temp = []
            len_temp = 0
if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

## 4) Time Delta

```python
#!/bin/python3

from datetime import datetime, timedelta

def time_delta(t1, t2):
    # Define the format of the timestamp
    fmt = "%a %d %b %Y %H:%M:%S %z"

    # Parse the timestamps using the defined format
    dt1 = datetime.strptime(t1, fmt)
    dt2 = datetime.strptime(t2, fmt)

    # Calculate the absolute difference in seconds
    diff_seconds = int(abs((dt1 - dt2).total_seconds()))

    return diff_seconds

# Number of test cases
t = int(input().strip())

for _ in range(t):
    # Read the timestamps
    time1 = input().strip()
    time2 = input().strip()

    # Calculate and print the absolute difference in seconds
    result = time_delta(time1, time2)
    print(result)
```

Time Delta

## 5) Find angle MBC.

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

Find angle MBC

## 6) No idea

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Read input values
n, m = map(int, input().split())
arr = list(map(int, input().split()))
set_a = set(map(int, input().split()))
set_b = set(map(int, input().split()))

# Calculate happiness
happiness = 0

for num in arr:
    if num in set_a:
        happiness += 1
    elif num in set_b:
        happiness -= 1

# Print the final happiness
print(happiness)
```

No idea

## 7) Word order

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import OrderedDict

def word_count(words):
    word_dict = OrderedDict()

    for word in words:
        # If the word is not in the dictionary, add it with count 1
        if word not in word_dict:
            word_dict[word] = 1
        else:
            # If the word is already in the dictionary, increment its count
            word_dict[word] += 1

    return word_dict

# Read input
n = int(input())
word_list = [input().strip() for _ in range(n)]

# Count occurrences
word_counts = word_count(word_list)

# Output the results
print(len(word_counts))
print(*word_counts.values())
```

Word order

## 8) Compress the string.

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import groupby
for k, c in groupby(input()):
    print("(%d, %d)" % (len(list(c)), int(k)), end=' ')

```

Compress the string

## 9) Company logo

```python
#!/bin/python3

import math
import os
import random
import re
import sys

from collections import Counter

S = input()
S = sorted(S)
FREQUENCY = Counter(list(S))
for k, v in FREQUENCY.most_common(3):
    print(k, v)

```

Company logo

## 10) Piling up

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
def can_stack_cubes(test_cases):
    for cubes in test_cases:
        n = cubes[0]
        side_lengths = cubes[1]

        left = 0
        right = n - 1
        prev_cube = float('inf')

        while left <= right:
            # Choose the larger cube from the left or right end
            current_cube = max(side_lengths[left], side_lengths[right])

            # Check if it's not possible to stack the cubes
            if current_cube > prev_cube:
                print("No")
                break

            # Update previous cube and adjust pointers
            prev_cube = current_cube
            if side_lengths[left] >= side_lengths[right]:
                left += 1
            else:
                right -= 1

        else:
            # If the loop completes without a break, print "Yes"
            print("Yes")

# Read the number of test cases
t = int(input().strip())

# Read and store test cases
test_cases = []
for _ in range(t):
    _ = int(input().strip())  # ignoring the number of cubes
    side_lengths = list(map(int, input().split()))
    test_cases.append((_, side_lengths))

# Check if it's possible to stack cubes for each test case
can_stack_cubes(test_cases)
```

Piling Up

---

## 11) Triangular quest 2

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

Triangular quest2

---

## 12) Iterables & Iterators

```python

# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations

N = int(input())
LETTERS = list(input().split(" "))
K = int(input())

TUPLES = list(combinations(LETTERS, K))
CONTAINS = [word for word in TUPLES if "a" in word]

print(len(CONTAINS)/len(TUPLES))

```

iterables and iterators

---

## 13) Triangular quest

```python
for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank line also
    print((10**(i)//9)*i)

```

Triangle quest

---

## 14) Classes: dealing with complex number

```python
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

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')
```

Classes:dealing with complex number



## 15) Athlete sort

```python
#!/bin/python3

import math
import os
import random
import re
import sys

# Read the first input for rows and columns
n, m = map(int, input().split())

# Read the matrix of numbers
rows = [list(map(int, input().split())) for _ in range(n)]

# Read the index for sorting
k = int(input())

# Sort rows based on the k-th column
for row in sorted(rows, key=lambda x: x[k]):
    print(' '.join(map(str, row)))

```

Athelete sort

---

## 16) Ginortx

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
def custom_sort(c):
    if c.islower():
        return (0, c)
    elif c.isupper():
        return (1, c)
    elif c.isdigit() and int(c) % 2 != 0:
        return (2, c)
    elif c.isdigit() and int(c) % 2 == 0:
        return (3, c)

def sort_string(s):
    sorted_str = ''.join(sorted(s, key=custom_sort))
    return sorted_str

# Read input
s = input().strip()

# Output the sorted string
result = sort_string(s)
print(result)

```

ginortS

---

## 17) Validating Email address with a filter

```python
def fun(email):
    try:
        username, url = email.split('@')
        website, extension = url.split('.')
    except ValueError:
        return False
    if username.replace('-', '').replace('_', '').isalnum() is False:
        return False
    elif website.isalnum() is False:
        return False
    elif len(extension) > 3:
        return False
    else:
        return True
def filter_mail(emails):
    return list(filter(fun, emails))
def filter_mail(emails):...
```

VALIDITY EMAIL ADDRESS

---

## 18) Reduce function.

```python
from fractions import Fraction...

def product(fracs):
    t = Fraction(reduce(lambda x, y: x * y, fracs))# complete this line with a reduce statement
    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)
```

Reduced function

---

## 19) Regrex substitution

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT

import re
for _ in range(int(input())):
    print(re.sub(r'(?<= )(&&|\|\|)(?= )', lambda x: 'and' if x.group() == '&&' else 'or', input()))
```

Regrex substitution

---

## 20) Validating Credit card number

```python

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

```

Validating credit card numbers

---

## 21) Word score

```python
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']
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

> ...
```

Word Score

---

## 22) Default argument

```python
class EvenStream(object):...

def print_from_stream(n, stream=EvenStream()):
    stream.__init__()
    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())
```

Default argument

---

# Hard task:

1) Maximize it

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Enter your code here. Read input from STDIN. Print output to STDOUT
import itertools

NUMBER_OF_LISTS, MODULUS = map(int, input().split())
LISTS_OF_LISTS = []

for i in range(0, NUMBER_OF_LISTS):
    new_list = list(map(int, input().split()))
    del new_list[0]
    LISTS_OF_LISTS.append(new_list)

def squared(element):
    return element**2

COMBS = list(itertools.product(*LISTS_OF_LISTS))
RESULTS = []

for i in COMBS:
    result1 = sum(map(squared, [a for a in i]))
    result2 = result1 % MODULUS
    RESULTS.append(result2)

print(max(RESULTS))
```

Maximize it

2) Validating postal codes

```python
regex_integer_in_range = r"^[1-9][\d]{5}$"    # Do not delete 'r'.
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)
```

Validity Postal codes

3) Matrix script

```python
#!/bin/python3

import math
import os
import random
import re
import sys
import re
n, m = map(int,input().split())
character_ar = ['''] * (n*m)
for i in range(n):
    line = input()
    for j in range(m):
        character_ar[i+(j*n)]=line[j]
decoded_str = ''.join(character_ar)
final_decoded_str = re.sub(r'(?<=[A-Za-z0-9])([ !@#$%&]+)(?=[A-Za-z0-9])',' ',decoded_str)
print(final_decoded_str)



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)

```

Matrix script

All Tasks were Completed on Hacker Net

## MEDIUM

| | | |
|---|---|---|
| **Write a function**<br>Medium, Python (Basic), Max Score: 10, Success Rate: 90.33% | ★ | Solved |
| **The Minion Game**<br>Medium, Python (Basic), Max Score: 40, Success Rate: 86.80% | ★ | Solved |
| **Merge the Tools!**<br>Medium, Problem Solving (Basic), Max Score: 40, Success Rate: 93.75% | ★ | Solved |
| **Time Delta**<br>Medium, Python (Basic), Max Score: 30, Success Rate: 91.36% | ★ | Solved |
| **Find Angle MBC**<br>Medium, Python (Basic), Max Score: 10, Success Rate: 89.15% | ★ | Solved |
| **No Idea!**<br>Medium, Python (Basic), Max Score: 50, Success Rate: 88.01% | ★ | Solved |
| **Word Order**<br>Medium, Python (Basic), Max Score: 50, Success Rate: 90.23% | ★ | Solved |
| **Compress the String!**<br>Medium, Python (Basic), Max Score: 20, Success Rate: 97.15% | ★ | Solved |
| **Company Logo**<br>Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.83% | ★ | Solved |
| **Piling Up!**<br>Medium, Python (Basic), Max Score: 50, Success Rate: 90.64% | ★ | 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: 94.11%
Solved

Validating Credit Card Numbers
Medium, Python (Basic), Max Score: 40, Success Rate: 95.46%
Solved

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

Default Arguments
Medium, Python (Intermediate), Max Score: 30, Success Rate: 78.82%
Solved

# HARD

| | | |
|---|---|---|
| **Maximize It!**<br>Hard, Problem Solving (Basic), Max Score: 50, Success Rate: 81.25% | ★ | Solved |
| **Validating Postal Codes**<br>Hard, Max Score: 80, Success Rate: 87.39% | ★ | Solved |
| **Matrix Script**<br>Hard, Problem Solving (Advanced), Max Score: 100, Success Rate: 89.97% | ★ | Solved |