# **COURSE: ARTIFICIAL INTELLIGENCE**

**ASSIGNMENT NO: 03** 



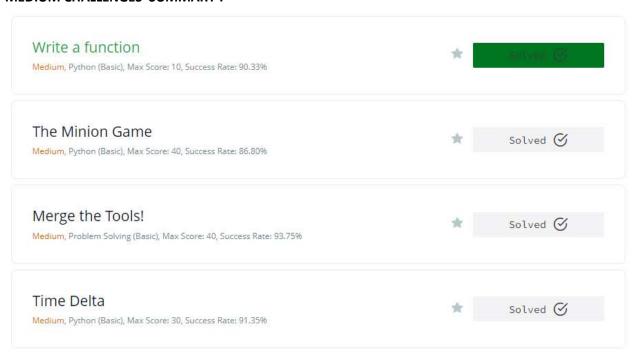
Submitted By
Mesum Ali Irfani
Regn Number
450291

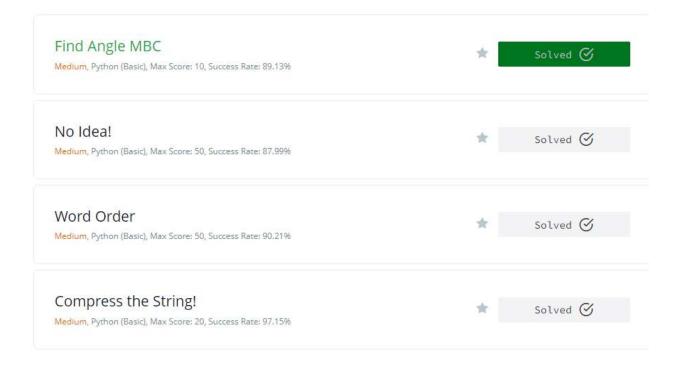
Submitted To Dr Yasar Ayaz

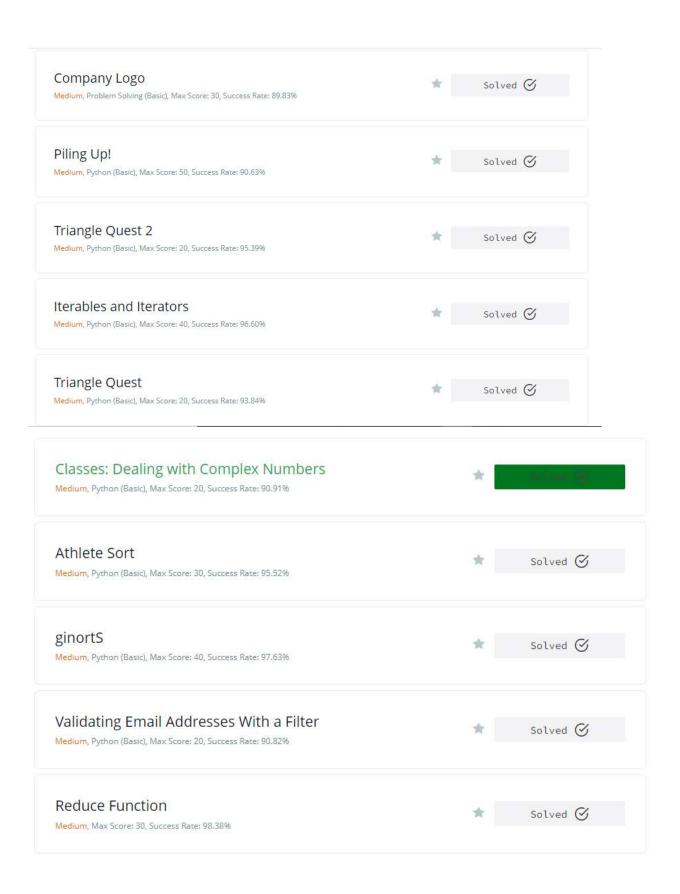
Department of Robotics and Artificial Intelligence

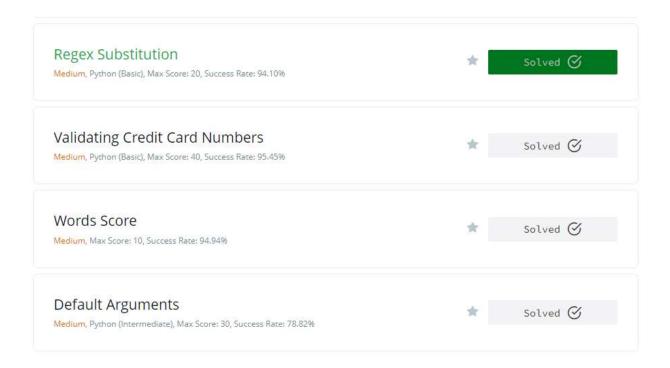
SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLO

#### **MEDIUM CHALLENGES SUMMARY:**









## **TASKS:**

# **TASK 1:**

```
1  def is_leap(year):
2     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
3         return True
4     else:
5         return False
6
7 > year = int(input())...
```

# **TASK 2:**

```
def minion_game(string):
 1
 2
         # your code goes here
 3
         vowels = "AEIOU"
 4
         stuart_score = 0
         kevin_score = 0
 5
         length = len(string)
 6
 7
 8
         for i in range(length):
 9
             if string[i] in vowels:
                 kevin_score += length - i
10
11
             else:
                 stuart_score += length - i
12
13
14
         if stuart_score > kevin_score:
             print("Stuart", stuart_score)
15
16
         elif kevin_score > stuart_score:
17
             print("Kevin", kevin_score)
18
         else:
19
             print("Draw")
                                                                   Line: 1 Col: 1
```

#### **TASK 3:**

```
Change Theme Language Python 3
     def merge_the_tools(string, k):
1
 2
         # your code goes here
 3
         n = len(string)
4
         for i in range(0, n, k):
5
             substring = string[i:i+k]
             unique_chars = ''
6
7
             for char in substring:
8
                 if char not in unique_chars:
9
                     unique_chars += char
             print(unique_chars)
11 > if __name__ == '__main__':...
```

### **TASK 4:**

import math

```
import os
import random
import re
import sys

# Complete the time_delta function below.
def time_delta(t1, t2):

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()
```

### **TASK 5:**

```
import math

def find_angle(opposite, adjacent):
    # Calculate the angle in radians using arctan
    angle_radians = math.atan(opposite / adjacent)

    # Convert the angle from radians to degrees
    angle_degrees = math.degrees(angle_radians)

    return round(angle_degrees) # Round to the nearest integer

# Read input lengths
side_a = float(input())
side_b = float(input())

# Calculate and print the angle
degree_symbol = chr(176)
result = find_angle(side_a, side_b)
print(f"{result}{degree_symbol}")
```

```
TASK 6:
```

```
if name == ' main ':
    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()))
    happiness = 0
    for num in arr:
        if num in set a:
            happiness += 1
        elif num in set b:
            happiness -= 1
    print(happiness)
TASK 7:
rom collections import OrderedDict
def word count(words):
    word dict = OrderedDict()
    for word in words:
        if word in word dict:
            word dict[word] += 1
        else:
            word dict[word] = 1
    return word dict
if __name__ == "__main__":
```

# **TASK 8:**

from itertools import groupby

print(len(word dict))

n = int(input().strip())

word dict = word count(words)

words = [input().strip() for in range(n)]

print(" ".join(map(str, word dict.values())))

```
def compress string(s):
    # Use groupby to group consecutive characters
    groups = [(len(list(g)), int(k)) for k, g in groupby(s)]
    # Format the result as specified
    result = " ".join(f"({count}, {char}))" for count, char in group
s)
    return result
# Read input string
input string = input()
# Compress the string and print the result
output string = compress string(input string)
print(output string)
TASK 9:
import math
import os
import random
import re
import sys
from collections import Counter
def print top characters(s):
    counter = Counter(s)
    sorted characters = sorted(counter.items(), key=lambda x: (-
x[1], x[0])
    for char, count in sorted characters[:3]:
        print(char, count)
if name == ' main ':
    s = input().strip()
   print top characters(s)
TASK 10:
from collections import deque
def can stack cubes(t, test cases):
    for test in test cases:
        n, cubes = test
        stack = deque(cubes)
        last cube = float('inf')
```

```
while stack:
            left, right = stack[0], stack[-1]
            if left >= right and left <= last cube:</pre>
                last cube = stack.popleft()
            elif right >= left and right <= last cube:</pre>
                last cube = stack.pop()
            else:
                print("No")
                break
        else:
            print("Yes")
if name == ' main ':
    t = int(input().strip())
    test cases = []
    for in range(t):
        n = int(input().strip())
        cubes = list(map(int, input().split()))
        test cases.append((n, cubes))
    can stack cubes(t, test cases)
TASK 11:
for i in range(1, int(input()) + 1):
    print(((10 ** i - 1) // 9) ** 2
TASK 12:
from itertools import combinations
def probability of a(n, letters, k):
    indices without a = [i for i, letter in enumerate(letters, star
t=1) if letter != 'a']
    total indices = n
    indices to select = k
    probability none contain a = len(list(combinations(indices with
out a, indices to select))) / \
                                  len(list(combinations(range(1, tot
al indices + 1), indices to select)))
   probability at least one contains a = 1 - probability none cont
ain a
    return round (probability at least one contains a, 4)
```

```
if name == " main ":
   n = int(input())
   letters = input().split()
   k = int(input())
    result = probability of a(n, letters, k)
   print(result)
TASK 13:
for i in range(1, int(input())):
    print((10 ** i - 1) // 9 * i)
TASK 15:
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())
    [print(*i) for i in sorted(arr, key=lambda attribute: attribute
[k])]
TASK 14:
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.ima
ginary)
    def sub (self, no):
       return Complex(self.real - no.real, self.imaginary - no.ima
ginary)
    def mul (self, no):
       real part = self.real * no.real - self.imaginary * no.imagi
nary
        imag_part = self.real * no.imaginary + self.imaginary * no.
real
       return Complex(real part, imag part)
    def truediv (self, no):
       den = no.real**2 + no.imaginary**2
       real part = (self.real * no.real + self.imaginary * no.imag
inary) / den
        imag part = (self.imaginary * no.real - self.real * no.imag
inary) / den
       return Complex(real part, imag part)
    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')
TASK 16:
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, c)
        else:
            return (3, c)
# Read input string
input string = input()
# Sort the string using the custom sorting key
sorted string = ''.join(sorted(input string, key=custom sort))
# Print the sorted string
print(sorted string)
TASK 17:
import re
def fun(s):
# Define the conditions for a valid email
    pattern = re.compile(r'^[\w-]+@[A-Za-z0-9]+\.[A-Za-z]\{1,3\}$')
    return bool(re.match(pattern, s))
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)
TASK 18:
from fractions import Fraction
from functools import reduce
def product(fracs):
    # Use reduce to find the product of all fractions
    result fraction = reduce(lambda x, y: x * y, fracs)
    return result fraction.numerator, result fraction.denominator
if name == ' main ':
    fracs = []
    for in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
   print(*result)
TASK 19:
import re
def modify symbols(text):
    # Replace '&&' with 'and' and '||' with 'or' with space on both
 sides
   modified text = re.sub(r'(?<=)&&(?=)', 'and', text)
   modified text = re.sub(r'(?<=) \setminus |(?=)', 'or', modified text)
   return modified text
if name == ' main ':
    # Read the number of lines
   n = int(input().strip())
    # Read and modify each line
    for in range(n):
        line = input().strip()
        modified line = modify symbols(line)
        print(modified line)
TASK 20:
import re
def is valid credit card(card number):
```

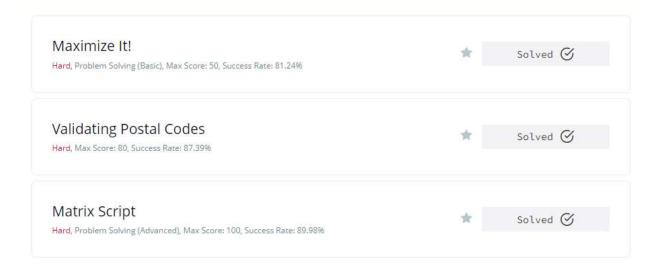
```
# Check if the card number matches the specified pattern
   pattern = re.compile(r'^[4-6]\d{3}-?\d{4}-?\d{4}-?\d{4}{\$'})
    if not re.match(pattern, card number):
        return "Invalid"
    # Remove hyphens from the card number
    card number = card number.replace('-', '')
    # Check for consecutive repeated digits
    consecutive repeated digits = re.compile(r'(\d)(\1{3,})')
    if re.search(consecutive repeated digits, card number):
        return "Invalid"
   return "Valid"
if name == " main ":
   n = int(input())
    for in range(n):
        card number = input().strip()
        result = is valid credit card(card number)
        print(result)
TASK 21:
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())
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())
TASK 22:
#!/bin/python3
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())
    [print(*i) for i in sorted(arr, key=lambda attribute: attribute
[k])]
```

#### HARD CHALLENGES



### **CODES OF THE TASKS**

#### **TASK 1: MAXIMIZE IT:**

```
from itertools import product

def maximize_value(n, m, lists):
    # Generate all possible combinations
    combinations = list(product(*lists))

    # Calculate the value for each combination and find the maximum
    max_value = max(sum(x**2 for x in combo) % m for combo in combi
nations)

    return max_value

# Input
n, m = map(int, input().split())
lists = [list(map(int, input().split()[1:])) for _ in range(n)]

# Output
result = maximize_value(n, m, lists)
print(result)
```

# TASK 2: Validating Postal Codes:

```
import re
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
TASK 3: MATRIX SCRIPT:
import re
def decode matrix script(rows, columns, matrix):
    # Transpose the matrix to read columns instead of rows
    transposed matrix = list(map(list, zip(*matrix)))
    # Concatenate alphanumeric characters from each column
    decoded script = ''.join([''.join(col) for col in transposed ma
trix1)
    # Replace symbols or spaces between alphanumeric characters wit
h a single space
    decoded script = re.sub(r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])
zA-Z0-9])', '', decoded script)
    return decoded script
# Input
rows, columns = map(int, input().split())
matrix = [input() for in range(rows)]
# Output
result = decode matrix script(rows, columns, matrix)
```

print(result)