PROBLEM 1

Introduction (1)

```
### PROBLEM 1 - Intro, Say "Hello World!" (1)

print("Hello, World!")
```

```
### PROBLEM 1 - Intro, Python if-else (2)

#!/bin/python3

if __name__ == '__main__':
    n = int(input().strip())
if n >= 1 and n <= 100:
    if n % 2 != 0:
        print("Weird")
    elif n % 2 == 0 and n >= 2 and n <=5:
        print("Not Weird")
    elif n % 2 == 0 and n >= 6 and n <= 20:
        print("Weird")
    elif n % 2 == 0 and n > 20:
        print("Not Weird")
```

```
### PROBLEM 1 - Intro, Arithmetic Operators (3)

a = 3
b = 5
print(a + b)
print(a - b)
print(a * b)
```

```
### PROBLEM 1 - Intro, Python Division (4)

a = 4
b = 5
print(a/b)
print(a//b)
```

```
### PROBLEM 1 - Intro, Loops (5)

if __name__ == '__main__':
```

```
    n = int(input("Enter a number: "))
for i in range(n):
    print(i ** 2)
```

```
### PROBLEM 1 - Intro, Write a function (6)

def is_leap(year):

    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

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

```
### PROBLEM 1 - Intro, Print a function (7)

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

    for i in range(1, n+1):
        print(i, end="")
```

Basic Data Types (2)

```
### PROBLEM 1 - Basic Data Types, List Comprehensions (1)

# Input for x, y, z, and n
x = int(input())
y = int(input())
z = int(input())
n = int(input())

coordinates = [[i, j, k] for i in range(x + 1) for j in range(y + 1) for k in range(z + 1

print(coordinates)
```

```
### PROBLEM 1 - Basic Data Types, Find the Runner-Up Score! (2)

if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
```

```
    unique_scores = sorted(set(arr), reverse=True)

    runner_up_score = unique_scores[1]

    print(runner_up_score)
```

```
### PROBELM 1 - Basic Data Types, Nested Lists (3)

if __name__ == '__main__':
  names = []
  for i in range(int(input())):
        name = input()
        score = float(input())
        names.append([name, score])
  AllScores = sorted(set([score for name, score in names]))

  SecondLowestScores = AllScores[1]

  SecondLowestNames = ([name for name, score in names if score == SecondLowestScores])

  SecondLowestNames.sort()
  for SingleName in SecondLowestNames:
        print(SingleName)
```

```
x=12
print(bin(x))
```

```
### PROBLEM 1 - Basic Data Types, Finding the percentage (4)

if __name__ == '__main__':
    n = int(input())
    student_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        student_marks[name] = scores

    query_name = input()

    query_scores = student_marks[query_name]
    average = sum(query_scores) / len(query_scores)

    print(f"{average:.2f}")
```

```
### PROBLEM 1 - Basic Data Types, Lists (5)

if __name__ == '__main__':
    N = int(input())
    list_1 = []

    for _ in range(N):
        command = input().split()
```

```
        if command[0] == 'insert':
            list_1.insert(int(command[1]), int(command[2]))

        elif command[0] == 'print':
            print(list_1)

        elif command[0] == 'remove':
            list_1.remove(int(command[1]))

        elif command[0] == 'append':
            list_1.append(int(command[1]))

        elif command[0] == 'sort':
            list_1.sort()

        elif command[0] == 'pop':
            list_1.pop()

        elif command[0] == 'reverse':
            list_1.reverse()
```

```
### PROBLEM 1 - Basic Data Types, Tuples (6)
```

Strings (3)

```
### PROBLEM 1 - Strings, sWAP cASE (1)

def swap_case(s):
    return s.swapcase()

if __name__ == '__main__':
    s = input()
    result = swap_case(s)
    print(result)
```

```
### PROBLEM 1 - Strings, String Split and Join (2)

def split_and_join(line):
    return '-'.join(line.split())

if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)
```

```
### PROBLEM 1 - Strings, What's Your Name? (3)
```

```
def print_full_name(first, last):
    print(f"Hello {first} {last}! You just delved into python.")

if __name__ == '__main__':
    first_name = input()
    last_name = input()
    print_full_name(first_name, last_name)
```

```
### PROBLEM 1 - Strings, Mutations (4)

def mutate_string(string, position, character):
    return string[:position] + character + string[position+1:]

if __name__ == '__main__':
    s = input()
    i, c = input().split()
    s_new = mutate_string(s, int(i), c)
    print(s_new)
```

```
### PROBLEM 1 - Strings, Find a string (5)

def count_substring(string, sub_string):
    count = 0
    for i in range(len(string) - len(sub_string) + 1):
        if string[i:i+len(sub_string)] == sub_string:
            count += 1
    return count

if __name__ == '__main__':
    string = input().strip()
    sub_string = input().strip()

    count = count_substring(string, sub_string)
    print(count)
```

```
### PROBLEM 1 - Strings, String Validators (6)

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

    # Check for any alphanumeric characters
    print(any(c.isalnum() for c in s))

    # Check for any alphabetical characters
    print(any(c.isalpha() for c in s))

    # Check for any digits
    print(any(c.isdigit() for c in s))

    # Check for any lowercase characters
    print(any(c.islower() for c in s))
```

```
    # Check for any uppercase characters
    print(any(c.isupper() for c in s))
```

```
### PROBLEM 1 - Strings, Text Alignment (7)

# Top Pillars
for i in range(thickness+1):
    print((c*thickness).center(thickness*2)+(c*thickness).center(thickness*6))

# Middle Belt
for i in range((thickness+1)//2):
    print((c*thickness*5).center(thickness*6))

# Bottom Pillars
for i in range(thickness+1):
    print((c*thickness).center(thickness*2)+(c*thickness).center(thickness*6))

# Bottom Cone
for i in range(thickness):
    print(((c*(thickness-i-1)).rjust(thickness)+c+(c*(thickness-i-1)).ljust(thickness)).r
```

```
### PROBLEM 1 - Strings, Text Wrap (8)

import textwrap

def wrap(string, max_width):
    return textwrap.fill(string, max_width)

if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

```
### PROBLEM 1 - Strings, Designer Door Mat (9)

n, m = map(int, input().split())

for i in range(1, n, 2):
    print(('.|.' * i).center(m, '-'))

print('WELCOME'.center(m, '-'))

for i in range(n-2, 0, -2):
    print(('.|.' * i).center(m, '-'))
```

```
### PROBLEM 1 - Strings, String Formatting (10)

def print_formatted(number):
    width = len(bin(number)) - 2

    for i in range(1, number + 1):
        print(f"{i:{width}d} {i:{width}o} {i:{width}X} {i:{width}b}")
```

```
if __name__ == '__main__':
    n = int(input())
    print_formatted(n)
```

```
### PROBLEM 1 - Strings, Alphabet Rangoli (11)

def print_rangoli(size):
    alphabet = [chr(i) for i in range(97, 97 + size)]

    lines = []

    # Create the rangoli pattern
    for i in range(size):
        left = alphabet[size-1:i:-1]
        right = alphabet[i:size]
        line = "-".join(left + right)
        lines.append(line.center(4*size - 3, '-'))

    print("\n".join(lines[::-1] + lines[1:]))

if __name__ == '__main__':
    n = int(input())
    print_rangoli(n)
```

```
### PROBLEM 1 - Strings, Capitalize! (12)

#!/bin/python3

import math
import os
import random
import re
import sys

def solve(s):
    return ' '.join([word.capitalize() for word in s.split(' ')])

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    s = input()

    result = solve(s)

    fptr.write(result + '\n')

    fptr.close()
```

```
### PROBLEM 1 - Strings, The Minion Game (13)

def minion_game(string):
    n = len(string)
```

```
    kevin_score = 0
    stuart_score = 0

    for i in range(n):
        if string[i] in 'AEIOU':
            kevin_score += n - i
        else:
            stuart_score += n - i

    # Determine the winner
    if kevin_score > stuart_score:
        print(f"Kevin {kevin_score}")
    elif stuart_score > kevin_score:
        print(f"Stuart {stuart_score}")
    else:
        print("Draw")

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

```
### PROBLEM 1 - Strings, Merge the Tools! (14)

def merge_the_tools(string, k):
    # Loop through the string in chunks of size k
    for i in range(0, len(string), k):
        t = string[i:i+k]

        u = ""

        for char in t:
            if char not in u:
                u += char

        print(u)

if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

Sets (4)

```
### PROBLEM 1 - Sets, Introduction to Sets (1)

def average(array):
    all_heights = set(array)

    all_heights_sum = sum(all_heights)

    heights_lenght = len(all_heights)
```

```
    avg = all_heights_sum / heights_lenght

    return round(avg, 3)

if __name__ == '__main__':
    n = int(input())
    arr = list(map(int, input().split()))
    result = average(arr)
    print(result)
```

```
### PROBLEM 1 - Sets, Symmetric Difference (2)

# Read input
m = int(input())
m_set = set(map(int, input().split()))

n = int(input())
n_set = set(map(int, input().split()))

symmetric_diff = m_set.symmetric_difference(n_set)

for value in sorted(symmetric_diff):
    print(value)
```

```
### PROBLEM 1 - Sets, No Idea! (3)

n, m = map(int, input().split())
array = list(map(int, input().split()))

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

happiness = 0

for i in array:
    if i in A:
        happiness += 1
    elif i in B:
        happiness -= 1

print(happiness)
```

```
### PROBLEM 1 - Sets, Set .add() (4)

n = int(input())

country_stamps = set()

for _ in range(n):
    country = input().strip()
    country_stamps.add(country)
```

```
print(len(country_stamps))
```

```
### PROBLEM 1 - Sets, Set .discard(), .remove() & .pop() (5)



```

```
### PROBLEM 1 - Sets, Set .union() Operation (6)

n = int(input())
english_subscribers = set(map(int, input().split()))

b = int(input())
french_subscribers = set(map(int, input().split()))

at_least_one_subscription = english_subscribers.union(french_subscribers)

print(len(at_least_one_subscription))
```

```
### PROBLEM 1 - Sets, Set .intersection() Operation (7)

n = int(input())
english_subscribers = set(map(int, input().split()))

b = int(input())
french_subscribers = set(map(int, input().split()))

both_subscriptions = english_subscribers.intersection(french_subscribers)

print(len(both_subscriptions))
```

```
### PROBLEM 1 - Sets, Set .difference() Operation (8)

n = int(input())
english_subscribers = set(map(int, input().split()))

b = int(input())
french_subscribers = set(map(int, input().split()))

english_only = english_subscribers.difference(french_subscribers)

print(len(english_only))
```

```
### PROBLEM 1 - Sets, Set .symmetric_difference() Operation (9)

n = int(input())
english_subscribers = set(map(int, input().split()))

b = int(input())
french_subscribers = set(map(int, input().split()))


```

```
symmetric_diff = english_subscribers.symmetric_difference(french_subscribers)

print(len(symmetric_diff))
```

```
### PROBLEM 1 - Sets, Set Mutations (10)

n_A = int(input())
A = set(map(int, input().split()))

N = int(input())

for _ in range(N):
    operation, _ = input().split()

    other_set = set(map(int, input().split()))

    if operation == "intersection_update":
        A.intersection_update(other_set)
    elif operation == "update":
        A.update(other_set)
    elif operation == "symmetric_difference_update":
        A.symmetric_difference_update(other_set)
    elif operation == "difference_update":
        A.difference_update(other_set)

print(sum(A))
```

```
### PROBLEM 1 - Sets, The Captain's Room (11)

K = int(input())
room_numbers = list(map(int, input().split()))

captain_room = (sum(set(room_numbers)) * K - sum(room_numbers)) // (K - 1)

print(captain_room)
```

```
### PROBLEM 1 - Sets, Check Subset (12)

T = int(input())

for _ in range(T):
    num_A = int(input())
    A = set(map(int, input().split()))

    num_B = int(input())
    B = set(map(int, input().split()))

    print(A.issubset(B))
```

```
### PROBLEM 1 - Sets, Check Strict Superset (13)

A = set(map(int, input().split()))
```

```
n = int(input())

is_strict_superset = True

for _ in range(n):
    other_set = set(map(int, input().split()))
    if not (A.issuperset(other_set) and A != other_set):
        is_strict_superset = False
        break

print(is_strict_superset)
```

Collections (5)

```
### PROBLEM 1 - Collections, collections.Counter() (1)

X = int(input())
shoe_sizes = list(map(int, input().split()))

N = int(input())

earnings = 0

for _ in range(N):
    size, price = map(int, input().split())
    if size in shoe_sizes:
        earnings += price
        shoe_sizes.remove(size)

print(earnings)
```

```
### PROBLEM 1 - Collections, DefaultDict Tutorial (2)

from collections import defaultdict

n, m = map(int, input().split())

A = defaultdict(list)
for i in range(1, n + 1):
    word = input().strip()
    A[word].append(i)

for _ in range(m):
    word = input().strip()
    if word in A:
        print(' '.join(map(str, A[word])))
    else:
        print(-1)
```

```
### PROBLEM 1 - Collections, Collections.namedtuple() (3)

n = int(input())

columns = input().split()

marks_index = columns.index('MARKS')

total_marks = 0

for _ in range(n):
    student_data = input().split()
    total_marks += int(student_data[marks_index])

average_marks = total_marks / n

print(f"{average_marks:.2f}")
```

```
### PROBLEM 1 - Collections, Collections.OrderedDict() (4)

n = int(input())

items = {}
order = []

for _ in range(n):
    *item_name, price = input().split()
    item_name = " ".join(item_name)
    price = int(price)

    if item_name in items:
        items[item_name] += price
    else:
        items[item_name] = price
        order.append(item_name)

for item_name in order:
    print(item_name, items[item_name])
```

```
### PROBLEM 1 - Collections, Word Order (5)

n = int(input())

word_count = {}
order = []

for _ in range(n):
    word = input().strip()
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1
        order.append(word)
```

```
print(len(order))

print(' '.join(str(word_count[word]) for word in order))
```

```
### PROBLEM 1 - Collections, Collections.deque() (6)

from collections import deque

d = deque()

n = int(input())

for _ in range(n):
    command = input().split()
    operation = command[0]

    if operation == 'append':
        d.append(command[1])
    elif operation == 'appendleft':
        d.appendleft(command[1])
    elif operation == 'pop':
        d.pop()
    elif operation == 'popleft':
        d.popleft()

print(' '.join(d))
```

```
### PROBLEM 1 - Collections, Piling Up! (7)

T = int(input())

for _ in range(T):
    n = int(input())

    cubes = list(map(int, input().split()))

    left = 0
    right = n - 1

    top = float('inf')

    possible = True
    while left <= right:
        if cubes[left] >= cubes[right]:
            chosen = cubes[left]
            left += 1
        else:
            chosen = cubes[right]
            right -= 1

        if chosen > top:
            possible = False
```

```
            break

        top = chosen

    if possible:
        print("Yes")
    else:
        print("No")
```

```
### PROBLEM 1 - Collections, Company Logo (8)

from collections import Counter

if __name__ == '__main__':
    s = input().strip()

    char_count = Counter(s)

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

    for i in range(min(3, len(sorted_characters))):
        print(sorted_characters[i][0], sorted_characters[i][1])
```

Date and Time (6)

```
### PROBLEM 1 - Date and Time, Calendar Module (1)

import calendar
from datetime import datetime

month, day, year = map(int, input().split())

date_object = datetime(year, month, day)

day_name = calendar.day_name[date_object.weekday()]

print(day_name.upper())
```

```
### PROBLEM 1 - Date and Time, Time Delta (2)

from datetime import datetime

def time_delta(t1, t2):
    fmt = "%a %d %b %Y %H:%M:%S %z"

    dt1 = datetime.strptime(t1, fmt)
    dt2 = datetime.strptime(t2, fmt)

    delta = abs((dt1 - dt2).total_seconds())

    return str(int(delta))
```

```
    return str(int(delta))

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()
```

(Errors and) Exceptions (7)

```
### PROBLEM 1 - (Errors and) Exceptions, Exceptions (1)

if __name__ == '__main__':
    t = int(input())

    for _ in range(t):
        try:
            a, b = map(int, input().split())
            result = a // b
            print(result)
        except ZeroDivisionError as e:
            print("Error Code:", e)
        except ValueError as e:
            print("Error Code:", e)
```

Built-Ins (8)

```
### PROBLEM 1 - Built-Ins, Zipped! (1)



```

```
### PROBLEM 1 - Built-Ins, Athlete Sort (4)

#!/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().split())))

    k = int(input())

    arr.sort(key=lambda x: x[k])

    for i in arr:
        print(*i)
```

```
### PROBLEM 1 - Built-Ins, ginortS (6)

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

if __name__ == "__main__":
    s = input().strip()

    sorted_string = ''.join(sorted(s, key=custom_sort_key))

    print(sorted_string)
```

Python Functionals (9)

```
### PROBLEM 1 - Python Functionals, Map and Lambda Function (1)

cube = lambda x: x**3

def fibonacci(n):
    fib_seq = []
    a, b = 0, 1
    for _ in range(n):
        fib_seq.append(a)
```

```
        a, b = b, a + b
    return fib_seq

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

    print(list(map(cube, fibonacci(n))))
```

Regex and Parsing challenges (10)

```
### PROBLEM 1 - Regex and Parsing challenges, Detect Floating Point Number (1)

def is_float(n):
    try:
        if n.count('.') != 1:
            return False

        float(n)

        if n[-1] == '.':
            return False

        return True
    except ValueError:
        return False

if __name__ == '__main__':
    t = int(input().strip())

    for _ in range(t):
        n = input().strip()
        print(is_float(n))
```

```
### PROBLEM 1 - Regex and Parsing challenges, Re.split() (2)

import re

regex_pattern = r"[,.]"

print("\n".join(re.split(regex_pattern, input())))
```

```
### PROBLEM 1 - Regex and Parsing challenges, Group(), Groups() & Groupdict() (3)

import re

s = input().strip()

match = re.search(r'([a-zA-Z0-9])\1', s)

if match:
    print(match.group(1))
```

```
else:
    print(-1)
```

```
### PROBLEM 1 - Regex and Parsing challenges, Re.findall() & Re.finditer() (4)

import re

s = input().strip()

pattern = r'(?<=[qwrtypsdfghjklzxcvbnmQWRTYPSDFGHJKLZXCVBNM])[aeiouAEIOU]{2,}(?=[qwrtypsd

matches = re.findall(pattern, s)

if matches:
    for match in matches:
        print(match)
else:
    print(-1)
```

```
### PROBLEM 1 - Regex and Parsing challenges, Re.start() & Re.end() (5)

s = input().strip()
k = input().strip()

start = 0
found = False

while start < len(s):
    index = s.find(k, start)

    if index == -1:
        break

    print(f"({index}, {index + len(k) - 1})")
    found = True

    start = index + 1

if not found:
    print("(-1, -1)")
```

```
### PROBLEM 1 - Regex and Parsing challenges, Regex Substitution (6)

import re

n = int(input())

for _ in range(n):
    line = input()
    line = re.sub(r'(?<= )&&(?= )', 'and', line)
    line = re.sub(r'(?<= )\|\|(?= )', 'or', line)
    print(line)
```

```
### PROBLEM 1 - Regex and Parsing challenges, Validating Roman Numerals (7)

import re

regex_pattern = r"^M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$"

print(str(bool(re.match(regex_pattern, input()))))
```

```
### PROBLEM 1 - Regex and Parsing challenges, Validating phone numbers (8)

import re

n = int(input())

pattern = r'^[789]\d{9}$'

for _ in range(n):
    number = input().strip()
    if re.match(pattern, number):
        print("YES")
    else:
        print("NO")
```

```
### PROBLEM 1 - Regex and Parsing challenges, Validating and Parsing Email Addresses (9)

import email.utils
import re

email_pattern = r'^[a-zA-Z][\w\.-]*@[a-zA-Z]+\.[a-zA-Z]{1,3}$'

n = int(input())

for _ in range(n):
    name, email_address = email.utils.parseaddr(input())

    if re.match(email_pattern, email_address):
        print(email.utils.formataddr((name, email_address)))
```

```
### PROBLEM 1 - Regex and Parsing challenges, Hex Color Code (10)

import re

n = int(input())
for i in range(n):
    m = re.findall(r'#[0-9A-Fa-f]{6}(?=\S)|#[0-9A-Fa-f]{3}(?=\S)',input())
    if m:
        for i in range(len(m)):
            print(m[i])
```

```
### PROBLEM 1 - Regex and Parsing challenges, HTML Parser - Part 1 (11)
```

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(f"Start : {tag}")
        for attr, value in attrs:
            print(f"-> {attr} > {value if value is not None else 'None'}")

    def handle_endtag(self, tag):
        print(f"End   : {tag}")

    def handle_startendtag(self, tag, attrs):
        print(f"Empty : {tag}")
        for attr, value in attrs:
            print(f"-> {attr} > {value if value is not None else 'None'}")

n = int(input())
html_code = "\n".join(input() for _ in range(n))

parser = MyHTMLParser()
parser.feed(html_code)
```

```
### PROBLEM 1 - Regex and Parsing challenges, HTML Parser - Part 2 (12)

from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle_comment(self, data):
        if '\n' in data:
            print(">>> Multi-line Comment")
            print(data)
        else:
            print(">>> Single-line Comment")
            print(data)

    def handle_data(self, data):
        if data.strip():
            print(">>> Data")
            print(data)

html = ""
for i in range(int(input())):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

```
### PROBLEM 1 - Regex and Parsing challenges, Detect HTML Tags, Attributes and Attribute

from html.parser import HTMLParser
```

```
class MyHTMLParser(HTMLParser):

    def handle_starttag(self, tag, attrs):
        print(tag)
        for attr, value in attrs:
            print(f"-> {attr} > {value}")

    def handle_startendtag(self, tag, attrs):
        print(tag)
        for attr, value in attrs:
            print(f"-> {attr} > {value}")

    def handle_comment(self, data):
        pass

n = int(input())
html = ""
for _ in range(n):
    html += input().rstrip() + '\n'

parser = MyHTMLParser()

parser.feed(html)
```

```
### PROBLEM 1 - Regex and Parsing challenges, Validating UID (14)

import re

def is_valid_uid(uid):
    if len(uid) != 10:
        return False

    if len(re.findall(r'[A-Z]', uid)) < 2:
        return False

    if len(re.findall(r'[0-9]', uid)) < 3:
        return False

    if not uid.isalnum():
        return False

    if len(set(uid)) != len(uid):
        return False

    return True

t = int(input())

for _ in range(t):
    uid = input().strip()
    if is_valid_uid(uid):
        print("Valid")
```

```
    else:
        print("Invalid")
```

```
### PROBLEM 1 - Regex and Parsing challenges, Validating Credit Card Numbers (15)

import re

def is_valid_credit_card(card_number):
    pattern = r'^[4-6]\d{3}(-?\d{4}){3}$'

    if not re.match(pattern, card_number):
        return "Invalid"

    card_number_clean = card_number.replace("-", "")

    if re.search(r'(\d)\1{3,}', card_number_clean):
        return "Invalid"

    return "Valid"

n = int(input())

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

```
### PROBLEM 1 - Regex and Parsing challenges, Validating Postal Codes (16)

import re

regex_integer_in_range = r"^[1-9][0-9]{5}$"

regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)"

P = input()

print (bool(re.match(regex_integer_in_range, P))
       and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

```
### PROBLEM 1 - Regex and Parsing challenges, Matrix Script (17)

#!/bin/python3

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)

print(re.sub(r'(?<=[A-Za-z0-9])([ !@#$%&]+)(?=[A-Za-z0-9])',' ', ''.join(s[i] for i in ra
```

XML (11)

```
### PROBLEM 1 - XML, XML 1 - Find the Score (1)

def get_attr_number(node):
    total_attr = len(node.attrib)

    if len(node) == 0:
        return total_attr
    return total_attr + sum(get_attr_number(child) for child in node)
```

```
### PROBLEM 1 - XML, XML2 - Find the Maximum Depth (2)

maxdepth = 0
def depth(elem, level):
    global maxdepth
    level += 1
    for child in elem:
        depth(child, level)
    maxdepth = max(maxdepth, level)
```

Closures and Decorators (12)

```
### PROBLEM 1 - Closures and Decorators, Standardize Mobile Number Using Decorators (1)

def wrapper(f):
    def fun(l):
        formatted_numbers = ['+91 {} {}'.format(number[-10:-5], number[-5:]) for number i
        return f(formatted_numbers)
    return fun

@wrapper
def sort_phone(l):
    print(*sorted(l), sep='\n')

if __name__ == '__main__':
    l = [input().strip() for _ in range(int(input()))]
    sort_phone(l)
```

```
### PROBLEM 1 - Closures and Decorators, Decorators 2 - Name Directory (2)

def person_lister(f):
    def inner(people):
        return [f(person) for person in sorted(people, key=lambda x: int(x[2]))]
    return inner

@person_lister
def name_format(person):
    return ("Mr. " if person[3] == "M" else "Ms. ") + person[0] + " " + person[1]

if __name__ == '__main__':
    people = [input().split() for i in range(int(input()))]
    print(*name_format(people), sep='\n')
```

NumPy (13)

```
### PROBLEM 1 - NumPy, Arrays (1)

import numpy

def arrays(arr):
    return numpy.array(arr[::-1], dtype=float)

arr = input().strip().split(' ')
result = arrays(arr)
print(result)
```

```
### PROBLEM 1 - NumPy, Shape and Reshape (2)

import numpy as np

arr = list(map(int, input().split()))

numpy_array = np.array(arr).reshape(3, 3)

print(numpy_array)
```

```
### PROBLEM 1 - NumPy, Transpose and Flatten (3)

import numpy as np

n, m = map(int, input().split())

matrix = np.array([input().split() for _ in range(n)], int)

print(np.transpose(matrix))

print(matrix.flatten())
```

```
### PROBLEM 1 - NumPy, Concatenate (4)

import numpy as np

n, m, p = map(int, input().split())

array_1 = np.array([input().split() for _ in range(n)], int)

array_2 = np.array([input().split() for _ in range(m)], int)

result = np.concatenate((array_1, array_2), axis=0)

print(result)
```

```
### PROBLEM 1 - NumPy, Zeros and Ones (5)

import numpy as np

shape = tuple(map(int, input().split()))

print(np.zeros(shape, dtype=int))

print(np.ones(shape, dtype=int))
```

```
### PROBLEM 1 - NumPy, Eye and Identity (6)

import numpy as np
np.set_printoptions(legacy='1.13')

n, m = map(int, input().split())

result = np.eye(n, m)

print(result)
```

```
### PROBLEM 1 - NumPy, Array Mathematics (7)

import numpy as np

n, m = map(int, input().split())

a = np.array([input().split() for _ in range(n)], int)
b = np.array([input().split() for _ in range(n)], int)

print(a + b)
print(a - b)
print(a * b)
print(a // b)
print(a % b)
print(a ** b)
```

```
### PROBLEM 1 - NumPy, Floor, Ceil and Rint (8)

import numpy as np

np.set_printoptions(legacy='1.13')

a = np.array(input().split(), float)

print(np.floor(a))
print(np.ceil(a))
print(np.rint(a))
```

```
### PROBLEM 1 - NumPy, Sum and Prod (9)

import numpy as np

n, m = map(int, input().split())
arr = np.array([input().split() for _ in range(n)], int)

sum_along_axis_0 = np.sum(arr, axis=0)
product_of_sum = np.prod(sum_along_axis_0)

print(product_of_sum)
```

```
### PROBLEM 1 - NumPy, Min and Max (10)

import numpy as np

n, m = map(int, input().split())
arr = np.array([input().split() for _ in range(n)], int)

min_along_axis_1 = np.min(arr, axis=1)
max_of_min = np.max(min_along_axis_1)

print(max_of_min)
```

```
### PROBLEM 1 - NumPy, Mean, Var, and Std (11)

import numpy as np

n, m = map(int, input().split())

arr = np.array([list(map(int, input().split())) for i in range(n)])

print(np.mean(arr, axis=1))
print(np.var(arr, axis=0))
print(round(np.std(arr), 11))
```

```
### PROBLEM 1 - NumPy, Dot and Cross (12)

import numpy as np
```

```
n = int(input())

a = np.array([list(map(int, input().split())) for _ in range(n)])

b = np.array([list(map(int, input().split())) for _ in range(n)])

result = np.dot(a, b)

print(result)
```

```
### PROBLEM 1 - NumPy, Inner and Outer (13)

import numpy as np

a = np.array(list(map(int, input().split())))

b = np.array(list(map(int, input().split())))

inner_product = np.inner(a, b)

outer_product = np.outer(a, b)

print(inner_product)
print(outer_product)
```

```
### PROBLEM 1 - NumPy, Polynomials (14)

import numpy as np

coefficients = list(map(float, input().split()))

x = float(input())

result = np.polyval(coefficients, x)

print(result)
```

```
### PROBLEM 1 - NumPy, Linear Algebra (15)

import numpy as np

n = int(input())

matrix = [list(map(float, input().split())) for _ in range(n)]

matrix_np = np.array(matrix)

determinant = np.linalg.det(matrix_np)

print(round(determinant, 2))
```

PROBLEM 2

```
##### PROBLEM 2 - Warmup, Birthday Cake Candles (1)

def birthdayCakeCandles(candles):
    tallest = max(candles)

    return candles.count(tallest)

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    candles_count = int(input().strip())

    candles = list(map(int, input().rstrip().split()))

    result = birthdayCakeCandles(candles)

    fptr.write(str(result) + '\n')

    fptr.close()
```

```
##### PROBLEM 2 - Implementation, Number Line Jumps (2)

def kangaroo(x1, v1, x2, v2):
    if v1 != v2:
        if (x2 - x1) % (v1 - v2) == 0 and (x2 - x1) / (v1 - v2) >= 0:
            return "YES"
    elif x1 == x2:
        return "YES"

    return "NO"

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    first_multiple_input = input().rstrip().split()

    x1 = int(first_multiple_input[0])
    v1 = int(first_multiple_input[1])
    x2 = int(first_multiple_input[2])
    v2 = int(first_multiple_input[3])

    result = kangaroo(x1, v1, x2, v2)

    fptr.write(result + '\n')
    fptr.close()
```

```
##### PROBLEM 2 - Implementation, Viral Advertising (3)

def viralAdvertising(n):
```

```
    shared = 5
    cumulative_likes = 0

    for day in range(1, n + 1):
        liked = shared // 2
        cumulative_likes += liked
        shared = liked * 3

    return cumulative_likes

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

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

    result = viralAdvertising(n)

    fptr.write(str(result) + '\n')
    fptr.close()
```

```
##### PROBLEM 2 - Sorting, Insertion Sort-Part 1 (4)

def insertionSort1(n, arr):
    value_to_insert = arr[-1]

    i = n - 2

    while i >= 0 and arr[i] > value_to_insert:
        arr[i + 1] = arr[i]
        print(*arr)
        i -= 1

    arr[i + 1] = value_to_insert
    print(*arr)
if __name__ == '__main__':
    n = int(input().strip())
    arr = list(map(int, input().rstrip().split()))
    insertionSort1(n, arr)
```

```
##### PROBLEM 2 - Sorting, Insertion Sort-Part 2 (5)

def insertionSort2(n, arr):
    for i in range(1, n):
        current_value = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > current_value:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = current_value
        print(*arr)

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

```
    arr = list(map(int, input().rstrip().split()))
    insertionSort2(n, arr)
```

```
##### PROBLEM 2 - Recursion, Recursive Digit Sum (6)

def superDigit(n, k):
    def recursive_super_digit(x):
        if len(x) == 1:
            return int(x)
        digit_sum = sum(int(digit) for digit in x)
        return recursive_super_digit(str(digit_sum))

    digit_sum_n = sum(int(digit) for digit in n)

    total_sum = digit_sum_n * k

    return recursive_super_digit(str(total_sum))

if __name__ == '__main__':
    first_multiple_input = input().rstrip().split()
    n = first_multiple_input[0]
    k = int(first_multiple_input[1])

    result = superDigit(n, k)
    print(result)
```