Python for Computational Problem-Solving

LABORATORY MANUAL

Week 3

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab session unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Input, Output, and Operators
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

18-09-2023 (Monday):
Sections - C, O, B, N

1) **Problem Statement:** Write a program that takes two integer inputs, representing the length and width of a rectangle, and calculates its area. Then, print the calculated area as output.
Solution:

```
length=int(input())
breadth=int(input())
print(length*breadth)
```

2) **Problem Statement:** Write a program that calculates the simple interest (SI) based on the given principal amount (P), interest rate (R), and time period (T). Use the formula SI = (P * R * T) / 100. Print the calculated simple interest as the output.
Solution:

```
p=float(input())
r=float(input())
t=float(input())
print(p*r*t/100)
```

3) **Problem Statement:** Write a Python program that prompts the user to enter three integer values. The program should then increment each of the input integers by 1 and print the results with a star symbol (*) in between them.

Solution:

```
num1 = int(input())
num2 = int(input())
num3 = int(input())

num1 = num1 + 1
num2 = num2 + 1
num3 = num3 + 1

print(num1 , "*" , num2 , "*" , num3)
```

4) **Problem Statement:** Create a program that prompts the user to input the radius of a circle. The program should then compute and display the area of the circle using the formula: Area = π * (radius^2), use the math module for π (pi) .

Solution:

```
import math
r = float(input())
print(math.pi * r *r)
```

5) **Problem Statement:**Build a program that requests the user to input their weight in kilograms and their height in meters. The program should then calculate and display their Body Mass Index (BMI) using the formula: BMI = weight / (height * height).

Solution:

```
weight_kg = float(input())
height_m = float(input())
bmi = weight_kg / (height_m ** 2)
print(bmi)
```

19-09-2023 (Tuesday):
Sections - F, A, D, R, M, P

6) **Problem Statement:** Take input as distance in meter, convert it to kilometer and display it.

Solution:

```
meter = int(input())
print(meter/1000)
```

7) **Problem Statement:** Write a program to calculate the area of a triangle by taking base(b) and height (h) as input. Use the formula Area of a triangle=½*b*h

Solution:

```
h = int(input())
b = int(input())
print(0.5*h*b)
```

8) **Problem Statement:** Write a Python program that takes three integer values as input from the user. The program should then decrease each of the input integers by 1 and print the results with a star hash (#) in between them.

Solution:

```
a = int(input())
b = int(input())
c = int(input())
a=a-1
b=b-1
c=c-1
print(a+'#'+b+'#'+c)
```

9) **Problem Statement:**Write a program that takes a string input from the user. The program should print the input string with a hash symbol (#) at the beginning and an exclamation mark (!) at the end.

Solution:

```
a = int(input())
print('#'+a+'!')
```

10) **Problem Statement:** Write a Program that takes a name as the input and prints a greeting message with the word "Hello" followed by the name enclosed in double quotes, such as 'Hello "John" '.

Solution:

```
a = input()
print('Hello', '\"' + a + '\"')
```

21-09-2023 (Thursday):
Sections - I, E,G, U, Q, S

11) **Problem Statement:** Write a program that takes input as currency in USD and converts it to INR. Use the formula 1 USD=82.9 INR.

Solution:

```
usd=int(input())
print(usd*82.9)
```

12) **Problem Statement:** Write a program to take price and quantity of a product that is bought as input and print the total cost. Use the formula Total cost = Price * Quantity.

Solution:

```
price=float(input())
quantity=int(input())
print(price*quantity)
```

13) **Problem Statement:** Write a program that takes the first and last name of the user as input and display the following message - The name is [last name], [first name] [last name].

Solution:

```
first = input()
last = input()
print("The name is " + last + ", " + first + " " + last)
```

14) **Problem Statement:** Write a program that takes date, month and year as input from the user and displays it in the format year/month/date

Solution:

```
date = int(input())
month = int(input())
year = int(input())
print(year, month, date, sep="/")
```

15) **Problem Statement:** Write a program that takes the number of days as input from the user and displays the number of years as an integer.

Solution:

```
days = int(input())
print(days // 365)
```

22-09-2023 (Friday):
Sections - J, V, K, W, H, T

16) **Problem Statement:** Write a program which takes two names as input and displays them with a comma in between them - [name1],[name2].

Solution:

```
name1 = input()
name2 = input()
print(name1, name2, sep=",")
```

17) **Problem Statement:** Write a program that takes two inputs, one in hours and the other in minutes, and display their sum in minutes.

Solution:

```
hours = int(input())
minutes = int(input())
print(hours*60+minutes)
```

18) **Problem Statement:** Write a Program that takes the user's name and their favourite colour as input and prints a message saying - 'Hello, [Name]! Your favourite colour is [Colour].'

Solution:

```
name = input()
favourite_colour = input()
print(f"Hello, {name}! Your favourite colour is {favourite_colour}.")
```

19) **Problem Statement:** Write a program that takes an integer as input and prints the remainder when it is divided by 2.

Solution:

```
num = int(input())
print(num % 2)
```

20) **Problem Statement:** Write a program that takes the name of a superhero as input and displays the message - 'I am [Superhero]'.

Solution:

```
superhero = input()
print("I am", superhero)
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 4

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program on combination of Control Structures
	• Program solutions for mandatory questions should be submitted for evaluation. • Competitive questions carry marks as part of the lab component, but it is optional for students to attempt.

(Submissions required. Students should submit their codes through hacker rank platform)

18-09-2023 (Monday):
Sections - C, O (11:30 AM to 1:00 PM):

1) **Problem Statement:** Write a program that takes a positive integer as input and calculates its factorial. The factorial of a positive integer n is the product of all positive integers from 1 to n.
Solution:

```
n = int(input())
factorial = 1

while n > 0:
    factorial *= n
    n -= 1
print(factorial)
```

2) **Problem Statement:** Write a program that takes a positive integer as input from the user and calculates the sum of its digits.
Solution:

```
num = int(input())
sum_of_digits = 0

while num > 0:
    digit = num % 10
    sum_of_digits += digit
    num //= 10
print(sum_of_digits)
```

3) **Problem Statement:** Write a program that takes a positive integer 'n' as input and counts the number of integers that are less than 'n' and divisible by both 3 and 5.
Solution:

```
n = int(input())
count = 0

for i in range(1, n):
    if i % 3 == 0 and i % 5 == 0:
        count += 1

print(count)
```

4) **(Competitive)Problem Statement:** Write a program that takes the number of rows 'n' as input from the user and prints a number pattern in increasing order as shown below.
Solution:

```
rows = int(input())
num = 1

for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print(num, end=" ")
        num += 1
    print()
```

18-09-2023 (Monday):
Sections - B, N (01:45 AM to 3:15 PM);

5) **Problem Statement:** Write a program that takes a student's score as input and prints their corresponding letter grade according to the following scale: A (90-100), B (80-89), C (70-79), D (60-69), F (0-59).

Solution:

```
score = int(input())

if score >= 90:
   print("A")
elif score >= 80:
   print("B")
elif score >= 70:
   print("C")
elif score >= 60:
   print("D")
else:
   print("F")
```

6) **Problem Statement:** Write a program that takes a positive integer as input and determines whether it is a prime number or not. Print 'True' if it's a prime number; otherwise, print 'False'. A prime number is a positive integer greater than 1 that is divisible only by 1 and itself.

Solution:

```
num = int(input())
is_prime = True

if num <= 1:
   is_prime = False
else:
   for i in range(2, int(num ** 0.5) + 1):
      if num % i == 0:
         is_prime = False
         break
print(is_prime)
```

7) **Problem Statement:** Write a program that takes a positive integer 'n' as input from the user and calculates the sum of all even numbers from 1 to 'n'.

Solution:

```
n = int(input())
total = 0
for i in range(2, n + 1, 2):
   total += i

print(total)
```

8) **(Competitive)Problem Statement:** Write a program that takes the number of rows 'n' as input from the user and prints a diamond pattern using asterisks (*) as shown in the sample example.. The pattern should have a diamond shape with 'n' rows in the middle.

Solution:

```
rows = int(input())
half_rows = (rows + 1) // 2

for i in range(1, half_rows + 1):
    print(" " * (half_rows - i) + "*" * (2 * i - 1))

for i in range(half_rows - 1, 0, -1):
    print(" " * (half_rows - i) + "*" * (2 * i - 1))
```

26-09-2023 (Tuesday):

Sections - F, A (8:00 to 10:30)

9) **Problem Statement:** Write a program that takes a student's score as input and prints their corresponding letter grade according to the following scale: A (90-100), B (80-89), C (70-79), D (60-69), F (0-59).

Solution:

```
score = int(input())

if score >= 90:
    print("A")
elif score >= 80:
    print("B")
elif score >= 70:
    print("C")
elif score >= 60:
    print("D")
else:
    print("F")
```

10) Problem Statement: Write a program that takes a positive integer as input from the user and calculates the sum of its digits.

Solution:

```
num = int(input())
sum_of_digits = 0

while num > 0:
   digit = num % 10
   sum_of_digits += digit
   num //= 10
print(sum_of_digits)
```

11) **Problem Statement: Problem Statement:** Write a Python program that prints a number pattern as follows: For a given positive integer n, the program should print n lines, where each line contains a sequence of digits equal to the line number, and that digit repeats as many times as the line number itself.

For example, if n is 4, the output should be:
1
22
333
4444

Solution:

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

12) **(Competitive) Problem Statement:** Write a program that takes the number of rows 'n' as input from the user and prints a number pattern in increasing order as shown below.

Solution:

```
rows = int(input())
num = 1
```

```
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print(num, end=" ")
        num += 1
    print()
```

26-09-2023 (Tuesday):

Sections - A, M (11:30 to 1:00)

13) **Problem Statement: Problem Statement:** Write a Python program that prints a number pattern as follows: For a given positive integer n, the program should print n lines, where each line contains a sequence of numbers from 1 to that line number. For example, if n is 4, the output should be:

1
12
123
1234

Solution:

```
n = int(input())
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(j, end='')
    print()
```

14) **Problem Statement:** Write a program to calculate the electricity bill from number of units based on following criteria:

First 10 units : No Charge
Next 15 units: Rs. 10/unit
Above 25 units: Rs. 25/unit

Solution:

```
units = int(input())
if units <= 10:
    print(0)
elif units <= 25:
    print((units - 10) * 10)
else:
    print((15 * 10) + (units - 25) * 25)
```

15) **Problem Statement: Problem Statement:** Write a program to print the count of digits in a given positive number.

Solution:

```
n=int(input())
if n==0:
   print(1)
else:
   count=0
   while n>0:
      count+=1
      n=n//10
   print(count)
```

16) **(Competitive) Problem Statement:** Write a program that calculates and prints the Least Common Multiple (LCM) of two positive integers entered by the user.

Solution:

```
num1 = int(input())
num2 = int(input())

if num1 == num2:
   print(num1)
else:
   max_num = max(num1, num2)
   lcm = max_num

   while True:
      if lcm % num1 == 0 and lcm % num2 == 0:
         break
      lcm += 1
   print(lcm)
```

26-09-2023 (Tuesday):
Sections - D, P (1:45 to 3:15)

17) **Problem Statement:** Write a program that given a month's name, prints the number of days in that month, assuming it is a non-leap year. For example, for "January," the program should output 31.

Solution:

```
month=input()
if month=='February':
   print(28)
elif month =='January' or month=='March' or month=='May' or month=='July' or month=='August' or month=='October' or month=='December':
   print(31)
else:
   print(30)
```

18) **Problem Statement:** Write a program that takes a positive integer as input and determines whether it is a prime number or not. Print 'True' if it's a prime number; otherwise, print 'False'. A prime number is a positive integer greater than 1 that is divisible only by 1 and itself.

Solution:

```
num = int(input())
is_prime = True

if num <= 1:
   is_prime = False
else:
   for i in range(2, int(num ** 0.5) + 1):
      if num % i == 0:
         is_prime = False
         break
print(is_prime)
```

19) **Problem Statement: Problem Statement:** Write a program to find the sum of the cube of all the even numbers from 1 to n.

Solution:

```
n=int(input())
sum=0
for i in range(2,n+1,2):
   sum+=i**3
print(sum)
```

20) **(Competitive) Problem Statement:** Write a program that takes the number of rows 'n' as input from the user and prints a diamond pattern using asterisks (*) as shown in the sample example.. The pattern should have a diamond shape with 'n' rows in the middle.

Solution:

```
rows = int(input())
half_rows = (rows + 1) // 2

for i in range(1, half_rows + 1):
   print(" " * (half_rows - i) + "*" * (2 * i - 1))

for i in range(half_rows - 1, 0, -1):
   print(" " * (half_rows - i) + "*" * (2 * i - 1))
```

28-09-2023 (Thursday):
Sections - I, U (8:00AM to 9:30AM)

21) **Problem Statement:** Write a program that given a month's name, prints the number of days in that month, assuming it is a non-leap year. For example, for "January," the program should output 31.

Solution:

```
month=input()
if month=='February':
   print(28)
elif month =='January' or month=='March' or month=='May' or month=='July' or month=='August' or month=='October' or month=='December':
   print(31)
else:
   print(30)
```

22) **Problem Statement:** Write a program to find the sum of the cube of all the even numbers from 1 to n.

Solution:

```
n=int(input())
sum=0
for i in range(2,n+1,2):
   sum+=i**3
print(sum)
```

23) **Problem Statement:** Write a Python program that prints a number pattern as follows: For a given positive integer n, the program should print n lines, where each line contains a sequence of digits equal to the line number, and that digit repeats as many times as the line number itself.

For example, if n is 4, the output should be:
1
22
333
4444

Solution:

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

24) **(Competitive) Problem Statement:** Write a program to check if a given number is an Armstrong number. An Armstrong number is a number that is equal to the sum of its own digits raised to the power of the number of digits. For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.

Solution:

```
n = int(input())
original = n
cnt = 0
result = 0
while n > 0:
```

```
    cnt += 1
    n = n // 10
n = original
while n > 0:
    digit = n % 10
    result += digit ** cnt
    n = n // 10
if result == original:
    print(True)
else:
    print(False)
```

28-09-2023 (Thursday):
Sections - E, Q (11:30AM to 1:00PM)

25) **Problem Statement:** Write a Python program that prints a number pattern as follows: For a given positive integer n, the program should print n lines, where each line contains a sequence of numbers from 1 to that line number.
For example, if n is 4, the output should be:
1
12
123
1234

Solution:

```
n = int(input())
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(j, end='')
    print()
```

26) **Problem Statement:** Write a program to calculate the electricity bill from number of units based on following criteria:
First 10 units : No Charge
Next 15 units: Rs. 10/unit
Above 25 units: Rs. 25/unit

Solution:

```
units = int(input())
if units <= 10:
   print(0)
elif units <= 25:
   print((units - 10) * 10)
else:
   print((15 * 10) + (units - 25) * 25)
```

27) **Problem Statement:** Write a program to print the count of digits in a given positive number.

Solution:

```
n=int(input())
if n==0:
   print(1)
else:
   count=0
   while n>0:
      count+=1
      n=n//10
   print(count)
```

28) **(Competitive) Problem Statement:** Write a program that calculates and prints the Least Common Multiple (LCM) of two positive integers entered by the user.

Solution:

```
num1 = int(input())
num2 = int(input())

if num1 == num2:
   print(num1)
else:
   max_num = max(num1, num2)
   lcm = max_num

   while True:
      if lcm % num1 == 0 and lcm % num2 == 0:
         break
      lcm += 1
   print(lcm)
```

28-09-2023 (Thursday):
Sections - G, S (1:45PM to 3:15PM)

29) **Problem Statement:** Write a program to print the count of odd numbers in a given range. The lower and upper range will be given as input.

Solution:

```
lower=int(input())
upper=int(input())
cnt=0
for i in range(lower,upper+1):
   if i%2!=0:
      cnt+=1
print(cnt)
```

30) **Problem Statement:** Write a program that takes an integer as input and prints the multiplication table for that number from 1 to 10.

Solution:

```
n=int(input())
for i in range(1,11):
   print(n,'*',i,'=',n*i)
```

31) **Problem Statement:** Write a Python program that prints all the numbers from 1 to n that are either multiples of 2 or multiples of 3.

Solution:

```
n=int(input())
for i in range(1,n+1):
   if i%2==0 or i%3==0:
      print(i)
```

32) **(Competitive) Problem Statement:** Write a Python program that checks if a given non-negative integer is a palindrome. A palindrome is a number that remains the same when its digits are reversed. For example, 121 is a palindrome because it reads the same forward and backward, whereas 1231 is not a palindrome.

Solution:

```
num=int(input())
if num < 0:
      print(False)
else:
   original_num = num
   reversed_num = 0

   while num > 0:
      digit = num % 10
      reversed_num = reversed_num * 10 + digit
      num = num // 10
print( original_num == reversed_num)
```

29-09-2023 (Friday):

Sections - J, V (8:30AM to 9:30AM)

33) **Problem Statement:** Write a program that accepts the length of three sides of a triangle and checks if it is an equilateral, isosceles or scalene triangle.

Solution:

```
a = int(input())
b = int(input())
c = int(input())
if a == b and b == c:
   print("Equilateral")
elif a == b or b == c or c == a:
   print("Isosceles")
else:
   print("Scalene")
```

34) **Problem Statement:** Write a program that prints all odd numbers between 1 and N (inclusive) that are divisible by 3 and 7.

Solution:

```
n = int(input())
for i in range(1, n+1, 2):
   if i % 3 == 0 and i % 7 == 0:
      print(i)
```

35) **Problem Statement:** Write a program to print the number of nonzero digits in a positive number.

Solution:

```
num = int(input())
count = 0
while num > 0:
    if num % 10 != 0:
        count += 1
    num = num // 10
print(count)
```

36) **(Competitive) Problem Statement:** Write a program that takes number of rows as input and prints the following pattern -

1
121
12321
1234321

Solution:

```
rows = int(input())
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print(j, end="")
    for k in range(i - 1, 0, -1):
        print(k, end="")
    print() # Move to the next line
```

29-09-2023 (Friday):
Sections - K, W (11:30AM to 1:00PM)

37) **Problem Statement:** Write a program that given a month's name, prints the number of days in that month. For example, for "January," the program should output 31.

Solution:

```
month = input()
if month == 'February':
   print(28)
elif month == 'April' or month == 'June' or month == 'September' or month == 'November':
   print(30)
else:
   print(31)
```

38) **Problem Statement:** Write a program that prints all the perfect squares between 1 and N (inclusive).

Solution:

```
n = int(input())
for i in range(1, n+1):
   # Taking integer values closest or equal to square root, truncates decimal
   sqt = int(i ** 0.5)
   if sqt ** 2 == i:
      print(i)
```

39) **Problem Statement:** Write a program to print the number of even (0 included) digits in a positive number.

Solution:

```
num = int(input())
count = 0
while num > 0:
   digit = num % 10
   if digit % 2 == 0:
      count += 1
   num = num // 10
print(count)
```

40) **(Competitive) Problem Statement:** Write a program that takes number of rows as input and prints the following pattern -

###1
##21
#321
4321

Solution:

```
rows = int(input())
for i in range(1, rows + 1):
    print("#" * (rows - i), end="")
    for j in range(i, 0, -1):
        print(j, end="")
    print()
```

29-09-2023 (Friday):

Sections - H, T (1:45PM to 3:15PM)

41) **Problem Statement:** Write a program to calculate the bonus an employee will receive from years of service based on the following criteria:
2 years or lesser : No bonus
Between 2 to 5 years: 3%
Greater than 5 years: 10%

Solution:

```
sal = int(input())
years = int(input())
if years <= 2:
    print("No bonus")
elif 2 < years <= 5:
    print(sal * 0.03)
else:
    print(sal * 0.10)
```

42) **Problem Statement:** Write a program that takes a positive number as input and prints the number in reverse.

Solution:

```
num = int(input())
reversed_num = 0
while num > 0:
    digit = num % 10
    reversed_num = reversed_num * 10 + digit
    num = num // 10
print(reversed_num)
    num = num // 10
print(count)
```

43) **Problem Statement:** Write a program that takes a positive number as input and pairwise prints all the factors of the number.

Solution:

```
n = int(input())
for i in range(1, int(n ** 0.5) + 1):
    if n % i == 0:
        # Floor division (//) to get integer values
        print(i, n // i)
```

44) **(Competitive)Problem Statement:** Write a program that takes number of rows as input and prints the following pattern -

1
10
101
1010

Solution:

```
rows = int(input())
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        if j % 2 == 1:
            print(1, end="")
        else:
            print(0, end="")
    print()
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 4

Topic: Control Structures

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Anchor and Lab Anchor – EC Campus: Prof. Kundhavai K R
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab session unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students to execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Input, Output, and Operators
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

Day: Monday
Sections: O2, C2, K2, B2, N2

Problem Statement 1:

Write a program to find the largest number among three numbers a,b,c. Take input of a,b,c and print the largest number on the stdout.

Solution:

```
# Input the three numbers
a = int(input("Enter the first number (a): "))
b = int(input("Enter the second number (b): "))
c = int(input("Enter the third number (c): "))
if a >= b and a >= c:
   largest = a
elif b >= a and b >= c:
   largest = b
else:
   largest = c
print("The largest number among", a, ",", b, ", and", c, "is", largest)
```

Problem Statement 2:

Write a python program to calculate sum of all even numbers from 1 to n. Take input n from stdin and print the sum on stdout.

Solution:

```
n = int(input("Enter a positive integer n: "))
even_sum = 0
for number in range(1, n + 1):
    # Check if the number is even
    if number % 2 == 0:
        even_sum += number  # Add even number to the sum
print("The sum of even numbers from 1 to", n, "is:", even_sum)
```

Problem Statement 3:

Write a Python program that takes input for the lengths of three sides of a triangle and determines if those values can form a triangle. If a triangle is possible, calculate and display its area, rounding it to two decimal places. Please provide the complete code for this task.

Solution:

```
a = int(input("Enter the length of the first side: "))
b = int(input("Enter the length of the second side: "))
c = int(input("Enter the length of the third side: "))

import math

if (a + b) <= c or (b + c) <= a or (c + a) <= b:
    print("It cannot form a triangle")
else:
    s = (a + b + c) / 2
    area = math.sqrt(s * (s - a) * (s - b) * (s - c))
    print("It can form a triangle")
    print("Area of the triangle is:", round(area, 2))
```

Problem Statement 4:

Write a Python program that takes input for marks obtained in five subjects and calculates the student's grade based on the following criteria:

If the average marks are between 81 and 100 (inclusive), assign the grade 'A'.
If the average marks are between 61 and 80 (inclusive), assign the grade 'B'.
If the average marks are between 41 and 60 (inclusive), assign the grade 'C'.
For any other average marks, assign the grade 'D'.

Solution:

```
subject1 = int(input("Enter marks for subject 1: "))
subject2 = int(input("Enter marks for subject 2: "))
subject3 = int(input("Enter marks for subject 3: "))
subject4 = int(input("Enter marks for subject 4: "))
subject5 = int(input("Enter marks for subject 5: "))

average = (subject1 + subject2 + subject3 + subject4 + subject5) / 5

if 81 <= average <= 100:
   grade = 'A'
elif 61 <= average <= 80:
   grade = 'B'
elif 41 <= average <= 60:
   grade = 'C'
else:
   grade = 'D'

print(f"Grade: {grade}")
```

Problem Statement 5:

Write a program to check whether a given date is valid. if yes, find the next date. Consider leap year as well.

Solution:

```
date = input("Enter the date in this format dd/mm/yyyy: ")
dd, mm, yy = date.split('/')
dd = int(dd)
mm = int(mm)
yy = int(yy)

if mm == 1 or mm == 3 or mm == 5 or mm == 7 or mm == 8 or mm == 10 or
mm == 12:
   max1 = 31
elif mm == 4 or mm == 6 or mm == 9 or mm == 11:
   max1 = 30
elif yy % 4 == 0 and yy % 100 != 0 or yy % 400 == 0:
   max1 = 29
else:
   max1 = 28
if mm < 1 or mm > 12:
   print("Date is invalid.")
elif dd < 1 or dd > max1:
   print("Date is invalid.")
else:
   if dd == max1 and mm != 12:
      dd = 1
      mm = mm + 1
   elif dd == 31 and mm == 12:
      dd = 1
      mm = 1
      yy = yy + 1
   else:
      dd = dd + 1

   print("Date is valid, and the next date is: {}/{}/{}".format(dd, mm, yy))
```

Day: Tuesday
Sections: D2, I2, E2, L2

Problem Statement 1:

Write a program to find the smallest number among three numbers a,b,c. Take input of a,b,c and print the smallest number on the stdout.

Solution:

```
# Input the three numbers
a = float(input("Enter the first number (a): "))
b = float(input("Enter the second number (b): "))
c = float(input("Enter the third number (c): "))

if a <= b and a <= c:
   smallest = a
elif b <= a and b <= c:
   smallest = b
else:
   smallest = c
print("The smallest number among", a, ",", b, ", and", c, "is", smallest)
```

Problem Statement 2:

Write a python program to calculate sum of all odd numbers from 1 to n. Take input n from STDIN and print the sum on STDOUT.

Solution:

```
n = int(input("Enter a positive integer n: "))

# Initialize a variable to store the sum of odd numbers
odd_sum = 0
# Loop through numbers from 1 to n and add odd numbers to the sum
for number in range(1, n + 1):
   if number % 2 != 0:
```

```
        odd_sum += number  # Add odd number to the sum
print("The sum of odd numbers from 1 to", n, "is:", odd_sum)
```

Problem Statement 3:

Write a program to calculate difference in area between two circles c1 and c2. Take input of radius r1 and r2.

Solution:

```
import math

r1 = float(input("Enter the radius of the first circle (c1): "))
r2 = float(input("Enter the radius of the second circle (c2): "))

area_c1 = math.pi * (r1 ** 2)
area_c2 = math.pi * (r2 ** 2)

area_difference = abs(area_c1 - area_c2)

print("The difference in area between the two circles is:",
round(area_difference, 2))
```

Problem Statement 4:

Implement a python program to calculate the LCM of two numbers.

Solution:

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
   greater = num1
else:
   greater = num2
while(True):
```

```
    if((greater % num1 == 0) and (greater % num2 == 0)):
        lcm = greater
        break
    greater += 1
print("LCM of",num1,"and",num2,"=",greater)
```

Problem Statement 5:

Write a program to find roots of a quadratic equation, ax^2 + bx + c = 0. What if b^2 - 4ac = 0, b^2 - 4ac > 0, b^2 - 4ac < 0. Print the type of roots as well.

Solution:

```
import math

print("Enter 3 coefficients:")
a = float(input("Enter the 1st coefficient: "))
b = float(input("Enter the 2nd coefficient: "))
c = float(input("Enter the 3rd coefficient: "))

determinant = b * b - 4 * a * c

# Condition for real and different roots
if determinant > 0:
    root1 = (-b + math.sqrt(determinant)) / (2 * a)
    root2 = (-b - math.sqrt(determinant)) / (2 * a)
    print("root1 =", root1, " and root2 =", root2)
    print("Real and distinct roots")

# Condition for real and equal roots
elif determinant == 0:
    root1 = root2 = -b / (2 * a)
    print("root1 =", root1, " and root2 =", root2)
    print("Real and equal roots")
# If roots are not real
else:
    realPart = -b / (2 * a)
    imaginaryPart = math.sqrt(-determinant) / (2 * a)
```

```
    print("root1 =", realPart, " + ", imaginaryPart, "i and root2 =", realPart, " - ",
imaginaryPart, "i")
    print("Imaginary roots")
```

Day: Wednesday
Sections: C1, K1, J2, F2

Problem Statement 1:

Write a Python program that asks the user to enter an integer. If the integer is even, print "Even," otherwise print "Odd."

Solution:

```
num = int(input("Enter an integer: "))
#condition for even
if num % 2 == 0:
    print("Even")
else:
    print("Odd")
```

Problem Statement 2:

Write a python code to print the first n multiples of a user given number .

Solution:

```
n = int(input("Enter the number of multiples: "))
m = int(input("Enter the base number: "))
for i in range(1, n + 1):
        multiple = m * i
        print(f"{m} x {i} = {multiple}")
```

Problem Statement 3:

Write a Python program to find the sum of all multiples of 3 or 5 below 100.

Solution:

```
total = 0
for i in range(1, 100):
    # condition for it to divisible by 3 or 5
    if i % 3 == 0 or i % 5 == 0:
        total += i
print("Sum of multiples of 3 or 5 below 100:", total)
```

Problem Statement 4:

Write a Python program to generate the Fibonacci sequence up to the nth term, where n is provided by the user.

Solution:

```
n = int(input("Enter the number of terms: "))
# Initializing the first two terms of the Fibonacci sequence
a, b = 0, 1

# Print the first two terms
print(a, end=' ')
print(b, end=' ')

# Generate and print the next 'n-2' terms
for _ in range(n - 2):
    # Calculate the next term
    next_term = a + b

    # Print the next term
    print(next_term, end=' ')

    # Update 'a' and 'b' for the next iteration
    a, b = b, next_term
print()
```

Problem Statement 5:

Write a python program using while loop and break statement to generate random numbers between 1 and 100 until it finds one that is divisible by both 7 and 9.

Solution:

```
import random

while True:
    number = random.randint(1, 100)

    if number % 7 == 0 and number % 9 == 0:
        print(f"Found a number ({number}) that is divisible by both 7 and 9.")
        break
    else:
        print(f"Generated number {number} is not divisible by both 7 and 9.
Trying again...")

print("Loop exited.")
```

Day: Thursday
Sections: B1, N1, O1, D1, M1

Problem Statement 1:

Write a Python program that takes a list of numbers as input and calculates the sum and average of the numbers.

Solution:

```
sum_of_numbers = 0
count = 0
numbers_input = input("Enter a list of numbers separated by spaces: ")
number_strings = numbers_input.split()
```

```
for num_str in number_strings:
    num = int(num_str)
    sum_of_numbers += num
    count += 1

# Check if there are numbers to avoid division by zero
if count > 0:
    average = sum_of_numbers / count
    print(f"Sum of numbers: {sum_of_numbers}")
    print(f"Average of numbers: {average}")
else:
    print("No numbers entered, cannot calculate sum and average.")
```

Using while loop:

```
total_sum = 0
count = 0
print("Enter a list of numbers (enter '0' to finish input):")
while True:
    number = float(input())

     if number == 0:
        break
    total_sum += number
    count += 1
if count > 0:
    average = total_sum / count
    print(f"Sum: {total_sum}")
    print(f"Average: {average}")
else:
    print("No numbers were entered.")
```

Problem Statement 2:

Write a Python program to calculate the factorial of a given number

Solution:

```
num = int(input("Enter a number: "))
factorial = 1
```

```
#keep in mind range() is always left inclusive and right exclusive
for i in range(1, num + 1):
    factorial *= i

print(f"Factorial of {num} is {factorial}")
```

Problem Statement 3:

Write a Python program to find and print all prime numbers between 10 and 50.

Solution:

```
# Define a range for prime number search (10 to 50)
start = 10
end = 50

print("Prime numbers between", start, "and", end, "are:")

# Loop through each number in the range
for num in range(start, end + 1):
    if num > 1:
        # Check for factors
        is_prime = True
        for i in range(2, int(num ** 0.5) + 1):
            if num % i == 0:
                is_prime = False
                break
        if is_prime:
            print(num)
```

Problem Statement 4:

Write a Python program to reverse a given number

Solution:

```
n = int(input("Enter a number: "))
```

```
reversed_num = 0

# Reverse the number
while n > 0:
    digit = n % 10
    reversed_num = reversed_num * 10 + digit
    n //= 10

print("Reversed number:", reversed_num)
```

Problem Statement 5:

Write a Python program to implement a simple calculator that can perform addition, subtraction, multiplication, and division of two numbers based on user input.

Solution:

```
print("Options:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

choice = input("Enter choice (1/2/3/4): ")

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if choice == '1':
    result = num1 + num2
    print("Result:", result)
elif choice == '2':
    result = num1 - num2
    print("Result:", result)
elif choice == '3':
    result = num1 * num2
    print("Result:", result)
```

```
elif choice == '4':
    if num2 == 0:
        print("Cannot divide by zero")
    else:
        result = num1 / num2
        print("Result:", result)
else:
    print("Invalid input")
```

Day: Friday
Sections: E1, L1, J1, F1, A1, I1

Problem Statement 1:

Write a program to find the sum of all numbers divisible by between 1 to n. Take input of n from stdin and print the sum on stdout

Solution:

```
# Input the value of n and m from the user
n = int(input("Enter a positive integer n: "))
m =3

divisible_sum = 0

for number in range(1, n + 1):
    if number % m == 0:
        divisible_sum += number

print(f"The sum of numbers divisible by 3 between 1 and {n} is:",
divisible_sum)
```

Problem Statement 2:

Write a program to check whether a given number is a prime number or not. Print yes if its else print no. Take input n from stdin and output it on stdout.

Solution:

```
n = int(input("Enter a positive integer n: "))

if n < 2:
   print("No")
else:
   is_prime = True
   for i in range(2, int(n**0.5) + 1):
      if n % i == 0:
         is_prime = False
         break

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

Problem Statement 3:

Write a program to find the GCD between two numbers.

Solution:

```
num1 = int(input("Enter the first integer: "))
num2 = int(input("Enter the second integer: "))

if num1 < num2:
   num1, num2 = num2, num1

while num2:
   num1, num2 = num2, num1 % num2

print("The GCD of the two numbers is:", num1)
```

Problem Statement 4:

Write a program to display sum of odd numbers and even numbers separately that fall between two numbers accepted from the user. (including both numbers) using while loop.

Solution:

```
num1 = int(input("Number1:"))
num2 = int(input("Number2:"))
sum_even = 0
sum_odd = 0
while num1 <= num2:
    if num1%2 == 0:
        sum_even = sum_even + num1
        num1 = num1 + 1
    else:
        sum_odd = sum_odd + num1
        num1 = num1 + 1
print("Even:",sum_even)
print("Odd:",sum_odd)
```

Problem Statement 5:

An atm has currency notes of denominations 10, 50 and 100.
Write a python program that calculates the number of hundred, fifty, and ten notes required to withdraw a given amount? Find the total number of currencies note the ATM disposes to the customer. – USE While loop.

Solution:

```
# Prompt the user for the amount to be withdrawn
amt = int(input("Enter the Amount to be Withdrawn: "))

# Initialize variables to count the number of each denomination
hundred = 0
fifty = 0
ten = 0
```

```
# Calculate the number of notes for each denomination
while amt >= 100:
   hundred += 1
   amt -= 100

while amt >= 50:
   fifty += 1
   amt -= 50

while amt >= 10:
   ten += 1
   amt -= 10

# Calculate the total number of currency notes
total_notes = hundred + fifty + ten

# Display the results
print("No of Hundred Notes:", hundred)
print("No of Fifty Notes:", fifty)
print("No of Ten Notes:", ten)
print("Total Number of Currency Notes Dispensed:", total_notes)
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 5

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Lists, Tuples and their combinations
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

18-09-2023 (Monday):
Sections - L, X (08:00 AM to 9:30 AM):

1) **Problem Statement:** Write a Python program that takes an integer 'n' as input and generates a list of squares for numbers from 1 to 'n'. After generating the list, print it. 'n' is inclusive.

Solution:

```
n = int(input())
square_list = [i ** 2 for i in range(1, n + 1)]
print(square_list)
```

2) **Problem Statement:** You are given a list containing the ages of employees in a company. Write a Python program that takes this list as input and performs the following tasks:
   - Converts the input list into a tuple.
   - Prints the tuple containing the ages of employees.
   - Finds and prints the sum of ages of the two youngest employees.
   - Finds and prints the sum of ages of the two oldest employees.

Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
```

```
    a = int(input())
    elements.append(a)

# Convert the list of elements into a tuple
employee_ages = tuple(elements)

# Print the tuple of employee ages
print(employee_ages)

# Sort the ages in ascending order
sorted_ages = sorted(employee_ages)

# Calculate the sum of ages of the two youngest employees
sum_of_youngest_ages = sum(sorted_ages[:2])

# Calculate the sum of ages of the two oldest employees
sum_of_oldest_ages = sum(sorted_ages[-2:])

# Print the sum of ages of the two youngest employees
print(sum_of_youngest_ages)

# Print the sum of ages of the two oldest employees
print(sum_of_oldest_ages)
```

3) **Problem Statement:** Write a Python program that takes a list of integers as input and counts the number of positive elements in the list.

Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    elements.append(a)

# Initialize a variable to count the positive elements
positive_count = 0
```

```
# Iterate through the list and count positive elements
for element in elements:
   if element > 0:
      positive_count += 1

# Print the count of positive elements
print(positive_count)
```

Bonus Question:

1) **Problem Statement:** You are given a square 2D list (matrix) representing a grid of integers. Write a Python program to calculate and print the sum of the elements along both the main diagonal (top-left to bottom-right) and the secondary diagonal (top-right to bottom-left) of the matrix.

   Solution:

```
# Read the size of the square matrix 'n'
n = int(input())

# Initialize a 2D list to store the matrix elements
matrix = []

# Read 'n' lines of input and create the matrix
for i in range(n):
   row = []
   for j in range(n):
      a = int(input())
      row.append(a)
   matrix.append(row)

# Initialize variables to store the sums of the main and secondary diagonals
main_diagonal_sum = 0
secondary_diagonal_sum = 0

# Calculate the sum of the elements along the main diagonal
for i in range(n):
   main_diagonal_sum += matrix[i][i]

# Calculate the sum of the elements along the secondary diagonal
for i in range(n):
   secondary_diagonal_sum += matrix[i][n - 1 - i]

# Print the sums of the main and secondary diagonals
```

```
print(main_diagonal_sum)
print(secondary_diagonal_sum)
```

18-09-2023 (Monday):
Sections - C, O (11:30 AM to 1:00 PM):

1) **Problem Statement:** Write a Python program that takes an integer 'n' as input and generates a list of squares for numbers from 1 to 'n'. After generating the list, print it. 'n' is inclusive.
   Solution:

```
n = int(input())
square_list = [i ** 2 for i in range(1, n + 1)]
print(square_list)
```

2) **Problem Statement:** Write a program that takes an integer 'x' as input, followed by an integer n, and then reads n integers to form a list. The program should count and return the number of occurrences of the specific element 'x' in the list.
   Solution:

```
# Read the integer 'x' to count
x = int(input())

# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    elements.append(a)

# Initialize a variable to count the occurrences
count = 0

# Iterate through the list and count occurrences of 'x'
for element in elements:
```

```
    if element == x:
        count += 1

# Print the count
print(count)
```

3) **Problem Statement:** You are given a list containing the ages of employees in a company. Write a Python program that takes this list as input and performs the following tasks:
   - Converts the input list into a tuple.
   - Prints the tuple containing the ages of employees.
   - Finds and prints the sum of ages of the two youngest employees.
   - Finds and prints the sum of ages of the two oldest employees.

   Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    elements.append(a)

# Convert the list of elements into a tuple
employee_ages = tuple(elements)

# Print the tuple of employee ages
print(employee_ages)

# Sort the ages in ascending order
sorted_ages = sorted(employee_ages)

# Calculate the sum of ages of the two youngest employees
sum_of_youngest_ages = sum(sorted_ages[:2])

# Calculate the sum of ages of the two oldest employees
sum_of_oldest_ages = sum(sorted_ages[-2:])

# Print the sum of ages of the two youngest employees
```

```
print(sum_of_youngest_ages)

# Print the sum of ages of the two oldest employees
print(sum_of_oldest_ages)
```

Bonus Question:

1) **Problem Statement:** You are given a 2D list (matrix) representing a grid of integers. Write a Python program to find and print the maximum value in the entire matrix along with its row and column indices.

Solution:

```
# Read the total number of rows 'rows' and columns 'cols'
rows = int(input())
columns = int(input())



# Initialize a 2D list to store the matrix elements
matrix = []

# Read 'rows' lines of input and create the matrix
for i in range(rows):
    row = []
    for j in range(cols):
        a = int(input())
        row.append(a)
    matrix.append(row)

# Initialize variables to store the maximum value and its indices
max_value = None
max_row = -1
max_col = -1

# Iterate through the matrix to find the maximum value and its indices
for i in range(rows):
    for j in range(cols):
        if max_value is None or matrix[i][j] > max_value:
            max_value = matrix[i][j]
```

```
        max_row = i
        max_col = j

# Print the maximum value and its indices
print(max_value)
print(max_row)
print(max_col)
```

18-09-2023 (Monday):
Sections - B, N (01:45 AM to 3:15 PM);

1) **Problem Statement:** Write a Python program that takes an integer 'n' as input and generates a list of squares for even numbers from 1 to 'n'. After generating the list, print it. 'n' is inclusive.

   Solution:

```
n = int(input())
square_even_list = [i ** 2 for i in range(1, n + 1) if i % 2 == 0]
print(square_even_list)
```

2) **Problem Statement:** Write a Python program that takes a list of integers as input and counts the number of positive elements in the list.
   Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
   a = int(input())
   elements.append(a)

# Initialize a variable to count the positive elements
```

```
positive_count = 0

# Iterate through the list and count positive elements
for element in elements:
   if element > 0:
      positive_count += 1

# Print the count of positive elements
print(positive_count)
```

3) **Problem Statement:** You are given a list of numbers. Your task is to write a Python program that performs the following steps using loops:

Read the total number of elements n.
Initialize an empty list to store the elements.
Read n integers and add them to the list.
Apply a custom hash function (element % 7) to every element in the list.
Convert the modified list into a tuple.
Print the tuple.
Calculate and print the average of all the elements in the tuple.

Refer to the Sample Question and Solution given below for more clarity.

Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
   a = int(input())
   elements.append(a)

# Initialize an empty list to store the modified elements
modified_elements = []

# Apply a custom hash function (element % 7) to every element in the list
```

```
for element in elements:
    modified_element = element % 7
    modified_elements.append(modified_element)

# Convert the modified list into a tuple
modified_tuple = tuple(modified_elements)

# Print the tuple of modified elements
print(modified_tuple)

# Calculate the average of all elements in the tuple
sum_of_elements = sum(modified_tuple)
average = sum_of_elements / len(modified_tuple)

# Print the average rounded to two decimal places
print(format(average,".2f"))
```

Bonus Question:

1) **Problem Statement:** You are given a square 2D list (matrix) representing a grid of integers. Write a Python program to calculate and print the sum of the elements along both the main diagonal (top-left to bottom-right) and the secondary diagonal (top-right to bottom-left) of the matrix.

   Solution:

```
# Read the size of the square matrix 'n'
n = int(input())

# Initialize a 2D list to store the matrix elements
matrix = []

# Read 'n' lines of input and create the matrix
for i in range(n):
    row = []
    for j in range(n):
        a = int(input())
        row.append(a)
    matrix.append(row)

# Initialize variables to store the sums of the main and secondary diagonals
```

```
main_diagonal_sum = 0
secondary_diagonal_sum = 0

# Calculate the sum of the elements along the main diagonal
for i in range(n):
   main_diagonal_sum += matrix[i][i]

# Calculate the sum of the elements along the secondary diagonal
for i in range(n):
   secondary_diagonal_sum += matrix[i][n - 1 - i]

# Print the sums of the main and secondary diagonals
print(main_diagonal_sum)
print(secondary_diagonal_sum)
```

10-10-2023 (Tuesday):

Sections - F, A (8:00AM to 10:30AM)

1) **Problem Statement:** Write a program that takes a list of integers and returns the count of the least frequently occurring element. Assume that only one element has the minimum frequency.

Solution:

```
n=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
if n==0 or n==1:
   print(n)
else:
   min_count = None
   least_frequent_element = None

   for item in l:
      count = l.count(item)
      if min_count is None:
         min_count=count
      elif count < min_count:
         min_count = count
         least_frequent_element = item
```

```
print(min_count)
```

2) **Problem Statement:** Write a program to convert a list into a sorted list in ascending order with only unique elements.

Solution:

```
n=int(input())
l=[]
for i in range(n):
    a=int(input())
    l.append(a)
unique=[]
for i in l:
    if i in unique:
        pass
    else:
        unique.append(i)
print(sorted(unique))
```

3) **Problem Statement:** Write a Python program to get the maximum of all the elements in a tuple within a specified range.

Solution:

```
n=int(input())
lower=int(input())
upper=int(input())
l=()
for i in range(n):
    a=int(input())
    l+=(a,)
max=lower-1
for i in l:
    if lower<=i<=upper:
        if i>max:
            max=i
if max!=lower-1:
```

```
    print(max)
else:
    print(None)
```

4) **(BONUS) Problem Statement:** You are given a 2D list (matrix) representing a grid of integers. Your task is to write a Python program to find and print the minimum value in the entire matrix along with its row and column indices.

Solution:

```
# Read the total number of rows 'rows' and columns 'cols'
rows = int(input())
columns = int(input())

# Initialize a 2D list to store the matrix elements
matrix = []

# Read 'rows' lines of input and create the matrix
for i in range(rows):
    row = []
    for j in range(cols):
        a = int(input())
        row.append(a)
    matrix.append(row)

# Initialize variables to store the minimum value and its indices
min_value = None
min_row = -1
min_col = -1

# Iterate through the matrix to find the minimum value and its indices
for i in range(rows):
    for j in range(cols):
        if min_value is None or matrix[i][j] < min_value:
            min_value = matrix[i][j]
            min_row = i
            min_col = j
```

```
# Print the minimum value and its indices
print(min_value)
print(min_row)
print(min_col)
```

10-10-2023 (Tuesday):

Sections - A, M (11:30AM to 1:00PM)

1) **Problem Statement:** Write a Python program to generate a list of squares for odd numbers from 1 to n and print the list.

   Solution:

```
n = int(input())
square_odd_list = [i ** 2 for i in range(1, n + 1) if i % 2 == 1]
print(square_odd_list)
```

2) **Problem Statement:** Given a list of integers, merge the second half of the list with the first half (in reverse order), and add the middle element to the first list if the total number of elements is odd.

   Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    elements.append(a)

# If 'n' is odd, add the middle element to the first list
if n % 2 != 0:
    middle_element = elements[n // 2]
    first_list = elements[:n // 2] + [middle_element]
    second_list = elements[n // 2 + 1:]
else:
    first_list = elements[:n // 2]
    second_list = elements[n // 2:]
```

```
# Merge the lists with second list first followed by first list
merged_list = second_list + first_list

# Print the merged list
print(merged_list)
```

3) **Problem Statement:** You are given a list of integers called num_list. Write a program that calculates and prints the average of all the elements present at odd indices in the list.

Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
num_list = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    num_list.append(a)

# Initialize variables for calculating the sum and count of elements at odd indices
odd_sum = 0
odd_count = 0

# Iterate through the list and calculate the sum and count of elements at odd indices
for i in range(1, n, 2):
    odd_sum += num_list[i]
    odd_count += 1

# Calculate the average of elements at odd indices
if odd_count != 0:
    average = odd_sum / odd_count
else:
    average = 0

# Print the average rounded to two decimal places
print(average)
```

4) **(BONUS) Problem Statement:** Write a program to delete all the elements which occur K times.

Solution:

```
n=int(input())
k=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
i=0
new=[]
for i in l:
   x=l.count(i)
   if x!=k:
      new.append(i)
l=new
print(l)
```

10-10-2023 (Tuesday):
Sections - D, P (1:45PM to 3:15PM)

5) **Problem Statement:** Write a Python program that takes an integer 'n' as input and generates a list of squares for numbers from 1 to 'n'. After generating the list, print it. 'n' is inclusive.

Solution:

```
n = int(input())
square_list = [i ** 2 for i in range(1, n + 1)]
print(square_list)
```

6) **Problem Statement:**You are given a list containing the ages of employees in a company. Write a Python program that takes this list as input and performs the following tasks:
   - Converts the input list into a tuple.
   - Prints the tuple containing the ages of employees.
   - Finds and prints the sum of ages of the two youngest employees.
   - Finds and prints the sum of ages of the two oldest employees.

Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    elements.append(a)

# Convert the list of elements into a tuple
employee_ages = tuple(elements)

# Print the tuple of employee ages
print(employee_ages)

# Sort the ages in ascending order
sorted_ages = sorted(employee_ages)

# Calculate the sum of ages of the two youngest employees
sum_of_youngest_ages = sum(sorted_ages[:2])

# Calculate the sum of ages of the two oldest employees
sum_of_oldest_ages = sum(sorted_ages[-2:])

# Print the sum of ages of the two youngest employees
print(sum_of_youngest_ages)

# Print the sum of ages of the two oldest employees
print(sum_of_oldest_ages)
```

7) **Problem Statement: Problem Statement:** Write a Python program that takes a list of integers as input and counts the number of positive elements in the list.

Solution:

```
# Read the total number of elements 'n'
n = int(input())

# Initialize an empty list to store the elements
```

```
elements = []

# Read 'n' integers and add them to the list
for i in range(n):
    a = int(input())
    elements.append(a)

# Initialize a variable to count the positive elements
positive_count = 0

# Iterate through the list and count positive elements
for element in elements:
    if element > 0:
        positive_count += 1

# Print the count of positive elements
print(positive_count)
```

8) **(BONUS) Problem Statement:**You are given a square 2D list (matrix) representing a grid of integers. Write a Python program to calculate and print the sum of the elements along both the main diagonal (top-left to bottom-right) and the secondary diagonal (top-right to bottom-left) of the matrix.

Solution:

```
# Read the total number of rows 'rows' and columns 'cols'
rows = int(input())
columns = int(input())


# Initialize a 2D list to store the matrix elements
matrix = []

# Read 'rows' lines of input and create the matrix
for i in range(rows):
    row = []
    for j in range(cols):
        a = int(input())
        row.append(a)
    matrix.append(row)
```

```
# Initialize variables to store the maximum value and its indices
max_value = None
max_row = -1
max_col = -1

# Iterate through the matrix to find the maximum value and its indices
for i in range(rows):
    for j in range(cols):
        if max_value is None or matrix[i][j] > max_value:
            max_value = matrix[i][j]
            max_row = i
            max_col = j

# Print the maximum value and its indices
print(max_value)
print(max_row)
print(max_col)
```

11-10-2023 (Wednesdayday):

Sections - Y (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program to get the HCF of the smallest and largest numbers in a list.

Solution:

```
n = int(input())
l = []
for i in range(n):
    a = int(input())
    l.append(a)

smallest = min(l)
largest = max(l)
```

```
hcf = 1
for i in range(smallest, 1, -1):
   if smallest%i==0 and largest%i==0:
      print(i)
      break
else:
   print(1)
```

2) **Problem Statement:** Given a list of integers, write a program that squares all the integers at even indices and doubles those at odd indices. Print the list thus obtained.

Solution:

```
n = int(input())
nums = []
for i in range(n):
   e = int(input())
   nums.append(e)

for i in range(n):
   if i % 2 == 0:
      nums[i] = nums[i] ** 2
   else:
      nums[i] = nums[i] * 2
print(nums)
```

3) **Problem Statement:** Given a list of strings, write a program to remove all four lettered words from it.

Solution:

```
n = int(input())
words = []
for i in range(n):
   w = input()
   words.append(w)

i = 0
while i < len(words):
   if len(words[i]) == 4:
      words.pop(i)
   else:
```

```
    i += 1
print(words)
```

Bonus Question:

4) **Problem Statement:** Write a program to get the kth smallest number from a list.

Solution:

```
n=int(input())
k=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
unique=[]
for i in l:
   if i in unique:
      pass
   else:
      unique.append(i)
unique.sort()
if  1 <= k <= len(unique):
   print(unique[k-1])
else:
   print(None)
```

12-10-2023 (Thursday):
Sections - I, U (8:00AM to 9:30 AM)

1) **Problem Statement:** Write a program that takes two tuples as input and returns a new list containing only the elements that are common between the two input lists.

Solution:

```
n = int(input())
m = int(input())
tuple1 = ()
for i in range(n):
   element = int(input())
   tuple1 += (element,)
tuple2 = ()
```

```
for i in range(m):
    element = int(input())
    tuple2 += (element,)
common_elements = ()
for element1 in tuple1:
    if element1 in tuple2:
        common_elements += (element1,)

print(common_elements)
```

2) **Problem Statement:** Write a program that takes a list of integers and returns the count of the most frequently occurring element. Assume that only one element has the maximum frequency.

Solution:

```
n=int(input())
l=[]
for i in range(n):
    a=int(input())
    l.append(a)
if n==0 or n==1:
    print(n)

else:
    max_count = 0
    most_frequent_element = None

    for item in l:
        count = l.count(item)
        if count > max_count:
            max_count = count
            most_frequent_element = item
    print(max_count)
```

3) **Problem Statement:** Write a program to print all elements from a list of positive integers that have exactly three digits and have 8 as digit in the hundreds place.

Solution:

```
n=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
for i in l:
   if 100 <= i <= 999 and (i // 100) % 10 == 8:
      print(i)
```

Bonus Question:

4) **Problem Statement:** Write a program to get the kth largest number from a list.

Solution:

```
n=int(input())
k=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
unique=[]
for i in l:
   if i in unique:
      pass
   else:
      unique.append(i)
unique.sort(reverse=True)
if  1 <= k <= len(unique):
   print(unique[k-1])
```

12-10-2023 (Thursday):
Sections - E, Q (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program that takes a list of integers as input and calculates the sum of all occurrences of the largest element in the list.

Solution:

```
n = int(input())
if n <= 0:
   print(0)
else:
   l = []
   for i in range(n):
      a = int(input())
      l.append(a)

   largest = max(l)
   sum_of_largest = largest * l.count(largest)
   print(sum_of_largest)
```

2) **Problem Statement:** Write a program to delete all the even elements of a list.

Solution:

```
n=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
i=0
while i<len(l):
   if l[i]%2==0:
      l.pop(i)
   else:
      i+=1
print(l)
```

3) **Problem Statement:** Write a Python program to get the average of all the elements in a tuple within a specified range.

Solution:

```
n=int(input())
lower=int(input())
```

```
upper=int(input())
l=()
for i in range(n):
   a=int(input())
   l+=(a,)
print(l)
sum=0
cnt=0
for i in l:
   if lower<=i<=upper:
      cnt+=1
      sum+=i
if cnt!=0:
   print(sum/cnt)
else:
   print(0.0)
```

Bonus Question:

4) **Problem Statement:** Write a program that modifies each element of a list by replacing it with the sum of the next two elements. Assume the list is circular, so the last element will be the sum of the elements at index[0] and index[1].

Solution:

```
n=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
result=[]
for i in range(len(l)):
   new = l[(i+1)%len(l)]+l[(i+2)%len(l)]
   result.append(new)
print(result)
```

12-10-2023 (Thursday):
Sections - G, S (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program that splits a list so that all elements at even indexes remain in the same list, while elements at odd indexes are moved to another list.

Solution:

```
n=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
new=l[1::2]
l=l[::2]
print(l,new,sep='\n')
```

2) **Problem Statement:** Write a program to get the LCM of the smallest and largest numbers in a list.

Solution:

```
n=int(input())
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
smallest=min(l)
largest=max(l)

lcm=largest
while True:
   if lcm % smallest == 0 and lcm % largest == 0:
      break
   lcm +=largest
print(lcm)
```

3) **Problem Statement:** Write a program to convert a list into a sorted list in descending order with only unique elements.

Solution:

```
n=int(input())
```

```
l=[]
for i in range(n):
   a=int(input())
   l.append(a)
unique=[]
for i in l:
   if i in unique:
      pass
   else:
      unique.append(i)
print(sorted(unique,reverse=True))
```

Bonus Question:

4) **Problem Statement:** You are given a square 2D list (matrix) representing a grid of integers. Your task is to write a program to calculate and print the average of the elements along the main diagonal of the matrix.

Solution:

```
m= int(input())
n = int(input())
matrix = []

for i in range(m):
   row = []
   for j in range(n):
      a = int(input())
      row.append(a)
   matrix.append(row)
main_diagonal_sum = 0
for i in range(n):
   main_diagonal_sum += matrix[i][i]
print(main_diagonal_sum/n)
```

13-10-2023 (Friday):

Sections - J, V (8:00AM to 9:30AM)

1) **Problem Statement:** Write a program to delete all the odd elements of a list. Print the new list thus formed.

Solution:

```
n = int(input())
l = []
for i in range(n):
   a = int(input())
   l.append(a)

i = 0
while i < len(l):
   if l[i] % 2 != 0:
      l.pop(i)
   else:
      i += 1

print(l)
```

2) **Problem Statement:** Write a Python program that takes a tuple of integers as input and returns a list containing the unique elements from the tuple.

Solution:

```
n = int(input())
nums = ()
for i in range(n):
   e = int(input())
   nums += (e,)
unique = []

for i in nums:
   if i not in unique:
      unique.append(i)

print(unique)
```

3) **Problem Statement:** Given a list of strings, write a program that counts the number of words having five or more letters.

Solution:

```
n = int(input())
```

```
words = []
for i in range(n):
   w = input()
   words.append(w)

cnt = 0
for i in words:
   if len(i) >= 5:
      cnt += 1

print(cnt)
```

Bonus Question:

4) **Problem Statement:** Given a matrix (2D list) of integers. Write a program to print the transpose of the matrix.

Solution:

```
n = int(input())
mat = []
for i in range(n):
   row = []  # Initialising every row

   for j in range(n):
      e = int(input())
      row.append(e)  # Adding each element to the row
   mat.append(row) # Adding each row to the matrix

for i in range(n):
   for j in range(i+1, n):
      mat[i][j], mat[j][i] = mat[j][i], mat[i][j]
print(mat)
```

13-10-2023 (Friday):
Sections - K, W (11:30AM to 1:00PM)

1) **Problem Statement:** Given a list of integers, write a program to print all elements that have exactly two digits.

Solution:

```
n = int(input())
l = []
for i in range(n):
   a = int(input())
   l.append(a)

new = []
for i in l:
   if 10 <= i <= 99 or -99 <= i <= -10:
      new.append(i)

print(new)
```

2) **Problem Statement:** Given a list of integers, write a program to swap adjacent elements in pairs (i.e., swap the first and second elements, the third and fourth elements, and so on). If the list has an odd number of elements, leave the last element in place. Print the list obtained after swapping.

Solution:

```
n = int(input())
nums = []
for i in range(n):
   e = int(input())
   nums.append(e)

for i in range(1, n, 2):
   nums[i], nums[i-1] = nums[i-1], nums[i]

print(nums)
```

3) **Problem Statement:** Write a program that takes a list of integers and returns two tuples. One containing all the even integers and one containing all odd integers in descending order

Solution:

```
n = int(input())
nums = []
for i in range(n):
```

```
    e = int(input())
    nums.append(e)

odd, even = (), ()
nums.sort(reverse=True)
for i in nums:
    if i % 2 == 0:
        even += (i,)
    else:
        odd += (i,)

print(odd, even, sep="\n")
```

Bonus Question:

4) **Problem Statement:** Write a program to delete all the elements which do not occur K times.

Solution:

```
n = int(input())
k = int(input())
l = []
for i in range(n):
    a = int(input())
    l.append(a)

new = []
for i in l:
    x = l.count(i)
    if x == k:
        new.append(i)

print(new)
```

13-10-2023 (Friday):
Sections - H, T (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program to get the HCF of the smallest and largest numbers in a list.

Solution:

```
n = int(input())
l = []
for i in range(n):
   a = int(input())
   l.append(a)

smallest = min(l)
largest = max(l)
hcf = 1
for i in range(smallest, 1, -1):
   if smallest%i==0 and largest%i==0:
      print(i)
      break
else:
   print(1)
```

2) **Problem Statement:** Given a list of integers, write a program that squares all the integers at even indices and doubles those at odd indices. Print the list thus obtained.

Solution:

```
n = int(input())
nums = []
for i in range(n):
   e = int(input())
   nums.append(e)

for i in range(n):
   if i % 2 == 0:
      nums[i] = nums[i] ** 2
   else:
      nums[i] = nums[i] * 2

print(nums)
```

3) **Problem Statement:** Given a list of strings, write a program to remove all four lettered words from it.

Solution:

```
n = int(input())
words = []
for i in range(n):
   w = input()
   words.append(w)

i = 0
while i < len(words):
   if len(words[i]) == 4:
      words.pop(i)
   else:
      i += 1

print(words)
```

Bonus Question:

4) **Problem Statement:** Write a program to get the kth smallest number from a list of integers.

Solution:

```
n = int(input())
k = int(input())
l = []
for i in range(n):
   a=int(input())
   l.append(a)

unique = []
for i in l:
   if i not in unique:
      unique.append(i)
unique.sort()

if  1 <= k <= len(unique):
   print(unique[k-1])
else:
   print(None)
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 5

Topic: Lists and Tuples

Semester: I
Course Code: UE23CS151A Course
Anchor: Prof. Sindhu R Pai Lab
Anchor: Dr. Divyashree N
Anchor and Lab Anchor – EC Campus: Prof. Kundhavai K R
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab session unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students to execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Input, Output, and Operators
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

Day: Monday
Sections: O2, C2, K2, B2, N2

Problem Statement 1:

Write a program that takes a tuple and an index as input and prints the element at that index.

Solution:

```
user_input = input("Enter a tuple of elements separated by spaces: ")
user_tuple = tuple(user_input.split())
index = int(input("Enter an index: "))
if 0 <= index < len(user_tuple):
  print(f"The element at index {index} is: {user_tuple[index]}")
else:
  print("Invalid index.")
```

Problem Statement 2:

Write a program that takes a list of numbers as input and calculates the sum of all elements in the list.

Solution:

```
user_input = input("Enter a list of numbers separated by spaces: ")

num_list = []
for num in user_input.split():
```

```
num = int(num)
  num_list.append(num)
sum_elements = sum(num_list)
print(f"The sum of elements is: {sum_elements}")
```

Problem Statement 3:

Write a Python program that takes two lists as input and returns a new list containing elements that are common in both input lists.

Solution:

```
# Take input lists from the user
list1 = input("Enter the first list (space-separated): ").split(' ')
list2 = input("Enter the second list (space-separated): ").split(' ')

# Convert the input strings to integer lists (assuming integers as
elements)
converted_list_1 = []
converted_list_2 = []

for item in list1:
# Strip leading and trailing whitespace from each item
    stripped_item = item.strip()

# Convert the stripped item to an integer and append it to the result
list
    converted_list_1.append(int(stripped_item))
for item in list2:
# Strip leading and trailing whitespace from each item
    stripped_item = item.strip()

# Convert the stripped item to an integer and append it to the result
list
converted_list_2.append(int(stripped_item))


# Find common elements
common_elements = []
for   value in
  converted_list_1: if value in
  converted_list_2:
common_elements.append(value)
```

```
# Print the result
print("Common elements:", common_elements)
```

Problem Statement 4:

Write a Python program that takes a tuple of numbers as input and performs the following operations:

1. Filter out all numbers greater than a specified threshold.
2. Square each of the filtered numbers.
3. Store the squared values in a new tuple.
4. Calculate and print the sum of the squared values.

Solution:

```
user_input = input("Enter a tuple of numbers separated by spaces: ")
user_tuple = tuple(map(float, user_input.split()))

# Input: Read a threshold value from the user
threshold = float(input("Enter a threshold value: "))

# Filter, square, and sum the numbers
filtered_and_squared = []

for num in user_tuple:
    # Check if the number is greater than the threshold
  if num > threshold:
      # Square the number and append it to the result list
    filtered_and_squared.append(num ** 2)



sum_of_squared_values = sum(filtered_and_squared)

# Output: Display the filtered and squared values, and the sum
print("Filtered and squared values:", filtered_and_squared)
print(f"Sum of squared values: {sum_of_squared_values:.2f}")
```

Problem Statement 5:

Write a program to find the addition of two matrices .

```
X = [[1,2,3],
[4 ,5,6],
[7 ,8,9]]
Y = [[9,8,7],
[6,5,4],
[3,2,1]]
result = [[0,0,0],
[0,0,0],
[0,0,0]]
for i in range(len(X)):
  for j in range(len(X[0])):
    result[i][j] = X[i][j] + Y[i][j]
for r in result:
  print(r)
```

Day: Tuesday

Sections: D2, I2, E2, L2, A2, M2

Problem Statement 1:

Write a python program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.

Solution:

```
values=input("Enter as many numbers you want (comma-separated): ")
str_list=values.split(",")
print(str_list)
l_num=[]
for str_num in str_list:
  l_num.append(int(str_num))
t_num=tuple(l_num)
print ("The Extracted values in list form are ",l_num)
print ("The Extracted values in tuple form are ",t_num)
```

Problem Statement 2:

Write a Python program that takes a value and an integer as input and creates a tuple containing the value repeated the specified number of times.

Solution:

```
# Input: Read a value and an integer from the user
value = input("Enter a value: ")
repetitions = int(input("Enter the number of times to repeat the value:
"))
# Create a tuple with the value repeated
result_tuple = (value,) * repetitions
# Output: Display the tuple
print("Resulting tuple:", result_tuple)
```

Problem Statement 3:

Write a Python program that swaps the values of two elements in a tuple given their indices.

Solution:

```
# Define a tuple
my_tuple = (1, 2, 3, 4, 5)

# Indices of elements to swap
index1 = 1
index2 = 3

# Convert the tuple to a list to make it mutable
temp_list = list(my_tuple)

# Swap the elements
temp_list[index1], temp_list[index2] = temp_list[index2],
temp_list[index1]

# Convert the list back to a tuple
my_tuple = tuple(temp_list)

# Print the result
print(my_tuple)
```

Problem Statement 4:

Write a Python program that takes a list of numbers as input.
The program should perform the following statistical calculations:
1.Calculate and print the mean (average) of the numbers in the list.
2. Calculate and print the median of the numbers in the list.
3. Find and print the mode (most frequently occurring number) in the list.

Solution:

```
# Take a list of numbers as input
b = input("Enter a list of numbers separated by spaces: ").split()
num_list=[]                #Fresh list
for i in b:
    num_list.append(int(i))
print(b)
# Calculate the mean (average)
mean = sum(num_list) / len(num_list)
print("Mean:", mean)

# Calculate the median
num_list.sort()
if len(num_list) % 2 == 0:
   middle1 = num_list[len(num_list) // 2]
   middle2 = num_list[len(num_list) // 2 - 1]
   median = (middle1 + middle2) / 2
else:
   median = num_list[len(num_list) // 2]
print("Median:", median)
# Calculate the mode
mode = None
max_count = 0

for num in num_list:
   count = num_list.count(num)
   if count > max_count:
      max_count = count
      mode = num

print("Mode:", mode)
```

Problem Statement 5:

Write a python program to find the sum of diagonal elements in nested 2D list.

Solution:

```
list_2D = [[1,2,3],[4,5,6],[7,8,9]]
sum = 0
for i in range(len(list_2D)):
 sum = sum + list_2D[i][i]
for row in list_2D:
 print(row)
print("sum of diagonal elements of nested 2D list is",sum)
```

Day: Wednesday
Sections: C1, K1, J2, F2

Problem Statement 1:

Write a python program to: find the frequency of largest element in a list. List may contain duplicates.

Solution:

```
List = [11, 22, 88, 53, 88, 53, 88]
largest = max(List) #using built-in function max
max_count = List.count(largest)
print("The frequncy of largest number", largest,"is",max_count)
```

Problem Statement 2:

Write a Python program that takes a tuple of elements and a specific element as input. The program should check and print whether the specified element exists in the tuple

Solution:

```
# Input: Read a tuple of elements and a specific element from the user
user_input = input("Enter a tuple of elements separated by spaces:
").split()
user_tuple = tuple(user_input)
element_to_search = input("Enter the element to search for: ")

# Check if the element exists in the tuple
if element_to_search in user_tuple:
    print(f"The element '{element_to_search}' exists in the tuple.")
else:
    print(f"The element '{element_to_search}' does not exist in the
tuple.")
```

Problem Statement 3:

Write a Python program that takes a list of numbers as input and remove duplicates from the list (do it with and without using sets)

Solution:

```
# Using sets
# Input a list of numbers from the user
input_list = input("Enter a list of numbers separated by spaces:
").split()

# Convert the input to a list of integers
for i in range(len(input_list)):
  input_list[i] = int(input_list[i])


# Remove duplicates using sets
unique_list = list(set(input_list))

# Print the list without duplicates
print("List without duplicates (using sets):", unique_list)
```

Problem Statement 4:

Write a Python program to flatten tuple of List to tuple.

Solution:

```
tuple_list = ([5, 6], [6, 7, 8, 9], [3])
print("The original tuple : ",tuple_list)
# Flatten tuple of List to tuple
res=[]
```

```
for lst in tuple_list:
 res.extend(lst)
res=tuple(res)
# printing result
print("The flattened tuple : " ,res)
```

```
# Wihtout using sets
# Input a list of numbers from the user
input_list = input("Enter a list of numbers separated by spaces:
").split()

# Convert the input to a list of integers
for i in range(len(input_list)):
input_list[i] = int(input_list[i])



# Create an empty list to store unique elements
unique_list = []

# Iterate through the input list and add unique elements to unique_list
for num in input_list:
    if num not in unique_list:
        unique_list.append(num)

# Print the list without duplicates
print("List without duplicates (without using sets):", unique_list)
```

Problem Statement 5:

Write a program to find the addition of two matrices .

```
X = [[1,2,3],
[4 ,5,6],
[7 ,8,9]]
Y = [[9,8,7],
[6,5,4],
[3,2,1]]
result = [[0,0,0],
[0,0,0],
[0,0,0]]
for i in range(len(X)):
  for j in range(len(X[0])):
    result[i][j] = X[i][j] + Y[i][j]
for r in result:
  print(r)
```

Day: Thursday
Sections: B1, N1, O1, D1, M1

Problem Statement 1:

Write a program that takes a list as input and reverses it in-place (with and without using built-in functions).

Solution:

```
user_input = input("Enter a list of elements separated by spaces: ")
input_list = user_input.split()
input_list.reverse()
print("Reversed list:", input_list)
print(input_list[::-1])
```

Problem Statement 2:

Write a Python program that takes a tuple of integers and a factor as input. The program should create a new tuple where each element is multiplied by the specified factor(E)

Solution:

```
# Input tuple of integers
input_tuple = (1, 2, 3, 4, 5)
```

```
# Input factor
factor = 2

# Create an empty list to store the multiplied elements
result_list = []

# Iterate through the input tuple
for element in input_tuple:
# Multiply each element by the factor and append to the result list
    result_list.append(element * factor)

# Convert the list to a tuple
result_tuple = tuple(result_list)

# Display the result
print("Original Tuple:", input_tuple)
print("Factor:", factor)
print("Resulting Tuple:", result_tuple)
```

Problem Statement 3:

Write a python program to find the union of two lists.(lists to be taken as user input)

Solution:

```
# Approach 1
# Input two lists of numbers from the user
list1 = input("Enter the first list of numbers separated by spaces:
").split()
list2 = input("Enter the second list of numbers separated by spaces:
").split()

# Convert the input to lists of integers
for i in range(len(list1)):
  list1[i] = int(list1[i])
for i in range(len(list2)):
list2[i] = int(list2[i])

# Find the union of the two lists
union_list = list(set(list1).union(set(list2)))

# Print the union list
```

```
print("Union of the two lists:", union_list)
```

```
#Approach 2
# Input two lists of numbers from the user
list1 = input("Enter the first list of numbers separated by spaces:
").split()
list2 = input("Enter the second list of numbers separated by spaces:
").split()

# Convert the input to lists of integers
for i in range(len(list1)):
  list1[i] = int(list1[i])
for i in range(len(list2)):
list2[i] = int(list2[i])

# Create an empty list to store the union
union_list = []

# Add unique elements from the first list to the union list
for num in list1:
    if num not in union_list:
        union_list.append(num)

# Add unique elements from the second list to the union list
for num in list2:
    if num not in union_list:
        union_list.append(num)

# Print the union list
print("Union of the two lists:", union_list)
```

Problem Statement 4:

Write a Python program to find the repeated items of a tuple.

Solution:

```
#create a tuple
num_tuple = (2, 4, 5, 6, 2, 3, 4, 4, 7)
num_set=set(num_tuple)
print(num_set)
for ele in num_set:
```

```
ele_count=num_tuple.count(ele)
 if ele_count>1:
print('The element',ele,'repeated',ele_count,'times')
```

Problem Statement 5:

Write a program to find the subtract of two matrices .

```
X = [[1,2,3],
[4 ,5,6],
[7 ,8,9]]
Y = [[9,8,7],
[6,5,4],
[3,2,1]]
result = [[0,0,0],
[0,0,0],
[0,0,0]]
for i in range(len(X)):
  for j in range(len(X[0])):
    result[i][j] = X[i][j] + Y[i][j]
for r in result:
  print(r)
```

Day: Friday
Sections: E1, L1, J1, F1, A1, I1

Problem Statement 1:

Write a python program to input a list a numbers and find the index of smallest element in a list.

Solution:

```
# Input a list of numbers from the user
num_list = input("Enter a list of numbers separated by spaces: ").split()

# Convert the input to a list of integers
for i in range(len(num_list)):
  num_list[i] = int(num_list[i])

# Initialize variables to store the minimum value
min_value = min(num_list)

print(f"Minimum value is at index {num_list.index(min_value)}")
```

Problem Statement 2:

Write a program that demonstrates tuple packing and unpacking. Create a tuple by packing multiple values into it, and then unpack the values into separate variables.

Solution:

```
# Tuple packing
my_tuple = ("John", 30, "Male")

# Tuple unpacking
name, age, gender = my_tuple
```

```
# Print the original tuple
print("Original Tuple:", my_tuple)

# Print the unpacked variables
print("Name:", name)
print("Age:", age)
print("Gender:", gender)
```

Problem Statement 3:

Write a python program to find the union of two lists.(lists to be taken as user input)

Solution:

Input two lists of numbers from the user
list1 = input("Enter the first list of numbers separated by spaces: ").split()
list2 = input("Enter the second list of numbers separated by spaces: ").split()

 # Convert the input to lists of integers for i in range(len(list1)):
list1[i] = int(list1[i]) for i in range(len(list2)):
list2[i] = int(list2[i])

 # Create an empty list to store the union union_list = []

 # Add unique elements from the first list to the union list for num in list1:
if num not in union_list: union_list.append(num)

 # Add unique elements from the second list to the union list for num in list2:
if num not in union_list: union_list.append(num)

 # Print the union list
print("Union of the two lists:", union_list)

Problem Statement 4:

Write a Python program that takes a tuple of numbers as input. The program should perform the following statistical calculations:

1. Calculate and print the mean (average) of the numbers in the tuple.
2. Calculate and print the median of the numbers in the tuple.
3. Find and print the mode (most frequently occurring number) in the tuple.

Solution:

```
# Input tuple of numbers
numbers = (5, 3, 2, 7, 2, 8, 4, 2)

# Calculate and print the mean (average)
mean_value = sum(numbers) / len(numbers)
print(f"Mean (Average): {mean_value}")

# Calculate and print the median
sorted_numbers = sorted(numbers)
n = len(sorted_numbers)
if n % 2 == 0:
median1 = sorted_numbers[n // 2 - 1]
    median2 = sorted_numbers[n // 2]
    median_value = (median1 + median2) / 2
else:
    median_value = sorted_numbers[n // 2]
print(f"Median: {median_value}")

# Calculate and print the mode
max_count = 0
mode_value = []

for num in numbers:
count = numbers.count(num) if
    count > max_count:
max_count = count
        mode_value = [num]
    elif count == max_count and num not in mode_value:
        mode_value.append(num)

print(f"Mode: {', '.join(map(str, mode_value))}")
```

Problem Statement 5:

Write a program to find the addition of two matrices.

```
X = [[1,2,3],
[4 ,5,6],
[7 ,8,9]]
Y = [[9,8,7],
[6,5,4],
[3,2,1]]
result = [[0,0,0],
[0,0,0],
[0,0,0]]
for i in range(len(X)):
  for j in range(len(X[0])):
    result[i][j] = X[i][j] + Y[i][j]
for r in result:
  print(r)
```

**

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 6

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Combination of Sets, Dictionaries and Strings using functions
	• Program solutions for mandatory questions should be submitted for evaluation. • Competitive questions carry marks as part of the lab component, but it is optional for students to attempt.

(Submissions required. Students should submit their codes through hacker rank platform)

16-10-2023 (Monday):
Sections - L,X (08:00 AM to 9:30 AM):

1) **Problem Statement:** Write a Python program that takes a string as input and counts the number of vowels and consonants in it. Ignore numbers and punctuation marks. Consider uppercase and lowercase letters as the same.
Solution:

```
# Prompt the user to enter a string
input_string = input()

# Define the vowels
vowels = "aeiouAEIOU"

# Convert the input string to lowercase to handle case insensitivity
input_string = input_string.lower()

# Initialize counters for vowels and consonants
vowel_count = 0
consonant_count = 0

# Loop through each character in the input string
for char in input_string:
    # Check if the character is in the list of vowels
    if char in vowels:
        vowel_count += 1
    # Check if the character is an alphabetic character (consonant)
    elif char.isalpha():
        consonant_count += 1
```

```
# Print the counts of vowels and consonants
print(vowel_count)
print(consonant_count)
```

2) **Problem Statement:** You are given a dictionary d, where each key-value pair represents a student's information, with the key being their Grade (a string) and the value being their Name. Write a Python program to calculate the number of students whose Grade is greater than or equal to B (A and B). If no one in the dictionary satisfies this condition, then print 0.The Grade in the Input can be in Uppercase or Lowercase.

Solution:

```
n = int(input())
d = {}
for i in range(n):
    name = str(input())
    grade = str(input())
    d[name] = grade

l = ["a","b","A","B"]
student = []
count = 0

for name, grade in d.items():
    if grade in l:
        student.append(name)

count = len(student)

print(count)
```

3) **Problem Statement:** You are given a set sets containing n distinct integer elements. Write a Python program to find the sum of the maximum and minimum elements in the set.

Solution:

```
n = int(input())
sets = set()
```

```
for i in range(n):
    a = int(input())
    sets.add(a)
#print(sets)

x = min(sets)
y = max(sets)
z = x+y
print(z)
```

Bonus Question:

1) **Problem Statement:** You are given a string test_str and a substring sub_str. Write a Python program to find the sub_str that occurs the most in a list of words formed by splitting test_str into words.

   Solution:

```
#maximum occurring substring from list
test_str = str(input())
sub_str = str(input())

max_count = 0
max_sub = ""

# split the input string into a list of words
words = test_str.split()

# loop over the words in the list
for word in words:
    # count the number of occurrences of the given substring in the word
    count = word.count(sub_str)
    # update max_count and max_sub if necessary
    if count >= max_count:
        max_count = count
        max_sub = sub_str * max_count

# printing result
print(max_sub)
```

18-09-2023 (Monday):

Sections - C, O (11:30 AM to 1:00 PM):

1) **Problem Statement:** You are given a list of integers. Your task is to write a Python program that calculates and prints the sum of unique integers in the list. Duplicate integers should not be considered in the sum.

Solution:

```
# Get the number of integers from the user
n = int(input())

# Initialize an empty set to store unique integers
set1 = set()

# Loop to input integers and add them to the set
for _ in range(n):
    element = int(input())
    # Add the integer to the set, sets automatically remove duplicates
    set1.add(element)

# Calculate the sum of unique integers using the sum() function
unique_sum = sum(set1)

print(unique_sum)
```

2) **Problem Statement:** You are given a dictionary containing student names as keys and their corresponding grades as values. Write a Python program that takes a grade threshold as input and displays the names of all the students who scored higher than that threshold in alphabetical order in a list.

Solution:

```
# Input the grade threshold
grade_threshold = float(input())

# Input the number of people
n = int(input())

# Initialize an empty dictionary to store student data
student_data = {}
```

```
# Input data for each person
for i in range(n):
    name = input()
    grade = float(input())
    student_data[name] = grade

# Initialize a list to store the names of students who met the criteria
high_graders = []

# Identify students who scored higher than the threshold
for name, marks in student_data.items():
    if marks > grade_threshold:
        high_graders.append(name)

# Sort the list of names alphabetically
high_graders.sort()

#print it
print(high_graders)
```

3) **Problem Statement:** Write a Python program that takes a string as input and counts the number of uppercase and lowercase letters in it. Your program should output the counts of uppercase and lowercase letters separately.

Solution:

```
# Prompt the user to enter a string
input_string = input()

# Initialize counters for uppercase and lowercase letters
uppercase_count = 0
lowercase_count = 0

# Loop through each character in the input string
for char in input_string:
    # Check if the character is an uppercase letter
    if char.isupper():
        uppercase_count += 1
    # Check if the character is a lowercase letter
```

```
    elif char.islower():
        lowercase_count += 1
    else:
        print(end="")
# Print the counts of uppercase and lowercase letters

print(uppercase_count)
print(lowercase_count)
```

Bonus Question:

1) **Problem Statement:** Write a Python program that checks if one string is a rotation of another string. For example, "waterbottle" is a rotation of "erbottlewat." Your program should print 'Y' if one string is a rotation of the other, and 'N' if it is not.
   Solution:

```
string1 = input()
string2 = input()

# Check if the lengths of both strings are the same
if len(string1) != len(string2):
    print("N")
else:
    combined_string = string1 + string1  # Concatenate the first string with itself
    if string2 in combined_string:
        print("Y")
    else:
        print("N")
```

18-09-2023 (Monday):
Sections - B, N (01:45 AM to 3:15 PM):

1) **Problem Statement:** Write a Python program that takes two sets as input and finds the number of common elements between them. Print the count of common elements.
   Solution:

```

# Input the number of elements for the first set and second set
n = int(input())
m = int(input())

# Initialize an empty set for the first set
```

```
set1 = set()
# Input elements for the first set and add them to set1
for _ in range(n):
    element = int(input())
    set1.add(element)

set2 = set()
for _ in range(m):
    element = int(input())
    set2.add(element)

# Calculate the intersection of set1 and set2 and store it in common_elements
common_elements = set1.intersection(set2)

print(len(common_elements))
```

2) **Problem Statement:** You are given a dictionary d, where each key-value pair represents a person's information, with the key being their age and the value being their salary. Write a Python program to calculate the average salary of all the people whose age is greater than 40. If there are no people older than 40, the program should output 0. Note: Don't include 40, it is exclusive.

Solution:

```
# Initialize an empty dictionary to store person's information
d = {}

# Input the number of people and their information
n = int(input())
for i in range(n):
    age = int(input())
    salary = float(input())
    d[age] = salary

# Initialize variables to keep track of total salary and count of people older than 40
total_salary = 0
count = 0

# Calculate the total salary of people older than 40
for age, salary in d.items():
    if age > 40:
```

```
        total_salary += salary
        count += 1

# Calculate and print the average salary (or 0 if no one is older than 40)
if count > 0:
    average_salary = total_salary / count
else:
    average_salary = 0

print(average_salary)
```

3) **Problem Statement:** Write a Python program that takes a string as input and converts the first half of the string to uppercase letters without changing the case of the second half of the string. If the string has an odd number of characters, include the middle character in the second half and do not convert it to uppercase.
Solution:

```
test_str = str(input())

#find the length of first half
hlf_idx = len(test_str) // 2

res = ""
for idx in range(len(test_str)):

    # uppercasing first half
    if idx < hlf_idx:
        res += test_str[idx].upper()
    else:
        res += test_str[idx]

# printing result
print(res)
```

Bonus Question:

1) **Problem Statement:** Write a Python program that takes a string as input and finds and prints all the unique substrings of the given string in a list.
Solution:

```
# Input a string
input_string = input()

# Initialize an empty set to store unique substrings
unique_substrings = set()

# Iterate through the string to find all substrings
for i in range(len(input_string)):
    for j in range(i + 1, len(input_string) + 1):
        substring = input_string[i:j]
        unique_substrings.add(substring)

# Print the unique substrings
print(sorted(list(unique_substrings)))
```

10-10-2023 (Tuesday):

Sections - F, A (8:00AM to 10:30AM)

1) **Problem Statement: Write a program that takes in a string as input, and returns all the unique letters in that string as a set**

   Solution:

```
# Get user input
s = input("Enter a string: ")

# Initialize an empty set to store unique alphabetic characters
letters_set = set()

# Convert the string to lowercase and iterate over it
s = s.lower()
for char in s:
    if char.isalpha():
        letters_set.add(char)

# Convert the set back to a sorted string and print it
sorted_letters = ''.join(sorted(letters_set))
print(sorted_letters)
```

2) **Problem Statement:** Write a program to check if one set is a subset of another.

Solution:

```
# Get the number of elements for set A and set B
n_A = int(input("How many elements in set A? "))
n_B = int(input("How many elements in set B? "))

# Get elements for set A
set_input_A = []
for i in range(n_A):
    element = input(f"Enter element {i+1} of set A: ")
    set_input_A.append(element)

# Get elements for set B
set_input_B = []
for i in range(n_B):
    element = input(f"Enter element {i+1} of set B: ")
    set_input_B.append(element)

# Convert the input lists to sets
A = set(set_input_A)
B = set(set_input_B)

# Check if A is a subset of B
is_subset = True
for item in A:
    if item not in B:
        is_subset = False
        break

print("A is a subset of B:", is_subset)
```

3) **Problem Statement:** Write a program to take in a list of letters, and then add them as keys to a dictionary, and their values should be the amount of times they appear in the list

Solution:

```
# Get the number of letters the user wants to input
n = int(input("How many letters do you want to input? "))

# Initialize an empty list to store the letters
letters = []

# Get each letter from the user
for i in range(n):
    letter = input(f"Enter letter {i+1}: ")
    letters.append(letter)

# Initialize an empty dictionary to store the letters and their counts
letter_counts = {}

# Iterate over the list of letters
for letter in letters:
    if letter in letter_counts:
        letter_counts[letter] += 1
    else:
        letter_counts[letter] = 1

# Print the dictionary
print(letter_counts)
```

4) **(BONUS) Problem Statement:** Write a program to check if one string input is an anagram of another

Solution:

```
# Get two strings from the user
str1 = input("Enter the first string: ")
str2 = input("Enter the second string: ")
```

```
# Remove spaces and convert to lowercase
str1 = str1.replace(" ", "").lower()
str2 = str2.replace(" ", "").lower()

# Check if sorted versions of the strings are the same
if sorted(str1) == sorted(str2):
   print("The strings are anagrams.")
else:
   print("The strings are not anagrams.")
```

10-10-2023 (Tuesday):

Sections - A, M (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program to count the number of vowels in a given string

Solution:

```
# Get a string from the user
s = input("Enter a string: ")

# Convert the string to lowercase for consistent comparison
s = s.lower()

# Define the vowels
vowels = 'aeiou'

# Count the vowels
count = 0
for char in s:
   if char in vowels:
      count += 1

# Print the result
print(f"The number of vowels in the string is: {count}")
```

2) **Problem Statement:** Write a program to demonstrate union and intersection operations in two inputted sets

Solution:

```
# Get the number of elements for the first set
n1 = int(input("How many elements in the first set? "))

# Initialize the first set and get the elements
set1 = set()
for i in range(n1):
    element = input(f"Enter element {i+1} of the first set: ")
    set1.add(element)

# Get the number of elements for the second set
n2 = int(input("How many elements in the second set? "))

# Initialize the second set and get the elements
set2 = set()
for i in range(n2):
    element = input(f"Enter element {i+1} of the second set: ")
    set2.add(element)

# Display the union and intersection
print("\nUnion of the two sets:", set1.union(set2))
print("Intersection of the two sets:", set1.intersection(set2))
```

3) **Problem Statement:** Write a program that merges two dictionaries. If there are common keys, add the values

Solution:

```
# Initialize two empty dictionaries
dict1 = {}
dict2 = {}

# Get the number of key-value pairs for the first dictionary
n1 = int(input("How many key-value pairs in the first dictionary? "))
```

```
# Collect the key-value pairs for the first dictionary
for i in range(n1):
    key = input(f"Enter key {i+1} of the first dictionary: ")
    value = int(input(f"Enter value for key {key}: "))
    dict1[key] = value

# Get the number of key-value pairs for the second dictionary
n2 = int(input("How many key-value pairs in the second dictionary? "))

# Collect the key-value pairs for the second dictionary
for i in range(n2):
    key = input(f"Enter key {i+1} of the second dictionary: ")
    value = int(input(f"Enter value for key {key}: "))
    dict2[key] = value

# Merge the dictionaries and add values for common keys
merged_dict = dict1.copy()  # Start with items from the first dictionary
for key, value in dict2.items():
    if key in merged_dict:
        merged_dict[key] += value  # If key is in dict1, add values
    else:
        merged_dict[key] = value  # Otherwise, add the key-value pair from dict2

# Display the merged dictionary
print("\nMerged dictionary:", merged_dict)
```

4) **(BONUS) Problem Statement:** Write a program to check if a string input is a palindrome

Solution:

```
# Get a string from the user
s = input("Enter a string: ")

# Remove spaces and convert to lowercase
cleaned_s = ""
for char in s.lower():
    if char.isalpha():
        cleaned_s += char
```

```
# Check if the string is a palindrome
is_palindrome = True
length = len(cleaned_s)
for i in range(length // 2):
    if cleaned_s[i] != cleaned_s[length - i - 1]:
        is_palindrome = False
        break

if is_palindrome:
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

17-10-2023 (Tuesday):

Sections - D, P (1:45PM to 3:15PM)

5) **Problem Statement:** You are given a string s. Write a Python program to reverse the string and then delete the last character of the reversed string to obtain a new string. Return the new string as the result.

Solution:

```
s = str(input())
rstr = s[::-1]
newstr = ''
for i in range(len(rstr)-1):
    newstr += rstr[i]
print(newstr)
```

6) **Problem Statement:** You are given a dictionary dicts, where each key-value pair represents an item's name (string) and its corresponding value (integer). Write a Python program to find the mean (average) of all the values in the dictionary.

Solution:

```
n = int(input())
dicts = {}

for i in range(n):
    a = str(input())
    b = int(input())
```

```
    dicts[a] = b

res = 0
for val in dicts.values():
    res += val

res = int(res / n)
print(res)
```

7) **Problem Statement: Problem Statement:** You are given a set ‘‘sets’’ containing n elements, which are strings. Write a Python program to count the number of vowels in the set elements. Consider only the English vowels, both uppercase and lowercase.

Solution:

```
n = int(input())
sets = set()

for i in range(n):
    a = str(input())
    sets.add(a)

vowel = "aeiouAEIOU"
count = 0
for word in sets:
    for alphabet in word:
        if alphabet in vowel:
            count = count + 1

print(count)
```

8) **(BONUS) Problem Statement:** Write a program in python that calculates the sum of the numerical values assigned to each letter in a given string, where 'a' is assigned the value 1, 'b' is assigned the value 2, and 'z' is assigned the value 26.

Solution:

```
# Initialize a variable to store the sum of letter values
letter_sum = 0
```

```
# Input string from stdin
input_string = input()

# Convert the input string to lowercase to handle both uppercase and lowercase letters
input_string = input_string.lower()

# Iterate through each character in the input string
for char in input_string:
    # Check if the character is a letter (a to z)
    if 'a' <= char <= 'z':
        # Calculate the numerical value of the letter and add it to the sum
        letter_value = ord(char) - ord('a') + 1
        letter_sum += letter_value

# Print the sum of letter values
print(letter_sum)
```

11-10-2023 (Wednesdayday):
Sections - Y (1:45PM to 3:15PM)

1) **Problem Statement:** You are given a dictionary containing student names as keys and their corresponding grades as values. Write a Python program that takes a grade threshold as input and displays the names of all the students who scored higher than that threshold in alphabetical order in a list.

   Solution:

```
# Input the grade threshold
grade_threshold = float(input())

# Input the number of people
n = int(input())

# Initialize an empty dictionary to store student data
student_data = {}
```

```
# Input data for each person
for i in range(n):
    name = input()
    grade = float(input())
    student_data[name] = grade

# Initialize a list to store the names of students who met the criteria
high_graders = []

# Identify students who scored higher than the threshold
for name, marks in student_data.items():
    if marks > grade_threshold:
        high_graders.append(name)

# Sort the list of names alphabetically
high_graders.sort()

#print it
print(high_graders)
```

2) **Problem Statement:** Write a Python program that takes two sets as input and finds the number of common elements between them. Print the count of common elements.
Solution:

```
# Input the number of elements for the first set and second set
n = int(input())
m = int(input())

# Initialize an empty set for the first set
set1 = set()
# Input elements for the first set and add them to set1
for _ in range(n):
    element = int(input())
    set1.add(element)

set2 = set()
for _ in range(m):
    element = int(input())
    set2.add(element)
```

```
# Calculate the intersection of set1 and set2 and store it in common_elements
common_elements = set1.intersection(set2)

print(len(common_elements))
```

3) **Problem Statement:** Write a Python program that takes a string as input and converts the first half of the string to uppercase letters without changing the case of the second half of the string. If the string has an odd number of characters, include the middle character in the second half and do not convert it to uppercase.

Solution:

```
test_str = str(input())

#find the length of first half
hlf_idx = len(test_str) // 2

res = ""
for idx in range(len(test_str)):

    # uppercasing first half
    if idx < hlf_idx:
        res += test_str[idx].upper()
    else:
        res += test_str[idx]

# printing result
print(res)
```

Bonus Question:

4) **Problem Statement:** Write a Python program that checks if one string is a rotation of another string. For example, "waterbottle" is a rotation of "erbottlewat." Your program should print 'Y' if one string is a rotation of the other, and 'N' if it is not.

Solution:

```
string1 = input()
string2 = input()

# Check if the lengths of both strings are the same
if len(string1) != len(string2):
    print("N")
```

```
else:
   combined_string = string1 + string1  # Concatenate the first string with itself
   if string2 in combined_string:
      print("Y")
   else:
      print("N")
```

19-10-2023 (Thursday):

Sections - I, U (8:00AM to 9:30 AM)

1) **Problem Statement:** Write a program that takes input for cities and their populations, stores them with city names as keys and population as values, and then identifies the city/cities with the highest population and prints the cities with maximum population in ascending order (alphabetical order).

Solution:

```
d = {}
n = int(input())
for i in range(n):
   city = input()
   population = int(input())
   d[city] = population
max_population = max(d.values())
max_list=[]
for i in d:
   if d[i]==max_population:
      max_list.append(i)
max_list.sort()
print('\n'.join(max_list))
```

2) **Problem Statement:** Write a program that takes a string, converts it to lower case and then calculates the frequency of each vowel in the string, and stores the results in a dictionary where each vowel is a key and the frequency is the corresponding value. Display the dictionary in ascending order of the vowels.

Solution:

```
vowel_frequency = {}

input_string = input()
```

```
input_string = input_string.lower()
vowels = "aeiou"
for char in input_string:
    if char in vowels:
        if char in vowel_frequency:
            vowel_frequency[char] += 1
        else:
            vowel_frequency[char] = 1
sorted_vowel_frequency = dict(sorted(vowel_frequency.items()))
for vowel, frequency in sorted_vowel_frequency.items():
    print(vowel,frequency)
```

3) **Problem Statement:** Write a program that takes two sets as input and calculates the difference between the two sets and then find the mean of the set obtained after the difference (rounded to 2 decimal places). If the difference is an empty set, print 0.

Solution:

```
n = int(input())
m = int(input())
set1 = set()
for _ in range(n):
    element = int(input())
    set1.add(element)
set2 = set()
for _ in range(m):
    element = int(input())
    set2.add(element)
difference_set = set1 - set2
if difference_set:
    mean_difference = sum(difference_set) / len(difference_set)
    print(format(mean_difference,'.2f'))
else:
    print(0)
```

Bonus Question:

4) **Problem Statement:** Write a program that takes a sentence as input and converts each alphabet in a given sentence to the next letter in the alphabet, while preserving the case of the letters. For example, a is converted to b, b to c, so on and z to a.

Solution:

```
input_sentence = input()
converted_sentence = ""
for char in input_sentence:
    if char.isalpha():
        if char == 'z':
            converted_sentence += 'a'
        elif char == 'Z':
            converted_sentence += 'A'
        else:
            converted_sentence += chr(ord(char) + 1)
    else:
        converted_sentence += char
print(converted_sentence)
```

19-10-2023 (Thursday):
Sections - E, Q (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program to store subject-wise marks for a student in a dictionary, where the subjects are keys and the corresponding marks are values. Print the dictionary as a list of tuples with subject and marks, sorted based on subject. Then, calculate and display the average marks for the student, rounded to two decimal places.

Solution:

```
subject_marks = {}
num_subjects = int(input())
for i in range(num_subjects):
    subject = input()
    marks = int(input())
    subject_marks[subject] = marks
```

```
total_marks = sum(subject_marks.values())
average_marks = total_marks / num_subjects
l=list(subject_marks.items())
l.sort(key=lambda x:x[0])
print(l)
print(format(average_marks,'.2f'))
```

2) **Problem Statement:** Write a program that takes a string as input and inverts the case of all characters in the string, converting lowercase letters to uppercase and uppercase letters to lowercase.

Solution:

```
input_string = input()
inverted_string = ""

for char in input_string:
    if char.islower():
        inverted_string += char.upper()
    elif char.isupper():
        inverted_string += char.lower()
    else:
        inverted_string += char
print(inverted_string)
```

3) **Problem Statement:** Write a program that takes string as input and prints all the words which have consecutive identical letters (double letters) in it. For example, hello has ll.

Solution:

```
input_sentence = input()
words = input_sentence.split()
words_with_double_consonants = []
for word in words:
    for i in range(len(word) - 1):
        if word[i].isalpha() and word[i + 1].isalpha() and word[i].lower() == word[i + 1].lower():
            words_with_double_consonants.append(word)
            break
if words_with_double_consonants:
```

```
print( '\n'.join(words_with_double_consonants))
```

Bonus Question:

4) **Problem Statement:** Write a program to take data type i.e string, integer or float as input and then the value as input in the same datatype. Store the values in a dictionary with key as input and value as datatype. Also, make a sentence with all the inputs which are of string datatype. For example, for the dictionary {"hello":"string", 5:"integer", "world":string} the sentence is "hello world".

Solution:

```
data_set = {}
n = int(input())
st=[]
for i in range(n):
    data_type = input()
    if data_type.lower() == "string":
        data_value = input()
        st.append(data_value)
    elif data_type.lower() == "integer":
        data_value = int(input())
    elif data_type.lower() == "float":
        data_value = float(input())
    data_set[data_value]=data_type

print(data_set)
if st:
    print(' '.join(st))
```

19-10-2023 (Thursday):
Sections - G, S (1:45PM to 3:15PM)

1) **Problem Statement:** Write a Python program to determine if two sets: set1 and set2, are disjoint (i.e., they have no common elements).

Solution:

```
n = int(input())
```

```
#create set1 with n elements
set1 = set()
for _ in range(n):
   element = int(input())
   set1.add(element)

m = int(input())
#create set2 with m elements
set2 = set()
for _ in range(m):
   element = int(input())
   set2.add(element)

# Check if the sets are disjoint
if not set1.intersection(set2):
   print("Disjoint sets")
else:
   print("Sets have common elements")
```

2) **Problem Statement:** Write a python code to create a dictionary whose keys are month names and whose values are the number of days in the corresponding months. Ask the user to enter a month name and use the dictionary to tell how many days are there in that month.

Solution:

```
month_days = { 'January': 31, 'February': 28, 'March': 31, 'April': 30, 'May': 31,
   'June': 30,  'July': 31, 'August': 31, 'September': 30, 'October': 31,
   'November': 30, 'December': 31 }

user_input = input("Enter a month name: ")
user_input = user_input.capitalize()

if user_input in month_days:
   days = month_days[user_input]
   print(f"{user_input} has {days} days.")
else:
   print("Enter a valid month name.")
```

3) **Problem Statement:** Write a program that reads a string and then prints a string that capitalizes every other letter in the string.

Solution:

```
 string = input()
length = len(string)
print ("Original string :", string)
string2 = " "

for a in range(0, length, 2):
      string2 += string[a]
      if a < (length-1) :
            string2 += string[a + 1].upper ()
print ("Alternatively capitalized string :" , string2)
```

Bonus Question:

4) **Problem Statement:** Write a python program that checks if two words are anagrams of each other. An anagram is a word or phrase formed by rearranging the letters of another.

Solution:

```
# Input words to be checked
sen1 = input()
sen2 = input()
sen1 = sen1.replace(" ", "").lower()
sen2 = sen2.replace(" ", "").lower()

# Check if the sorted characters in both words match
if sorted(sen1) == sorted(sen2):
   print("Anagrams")
else:
   print("Not Anagrams")
```

13-10-2023 (Friday):
Sections - J, V (8:00AM to 9:30AM)

1) **Problem Statement:** Given two sets, set1 and set2, write a Python program to find and print the common elements in ascending order.

Solution:

```
n = int(input())
#create set1 with n elements
set1 = set()
for _ in range(n):
   element = int(input())
   set1.add(element)

m = int(input())
#create set2 with m elements
set2 = set()
for _ in range(m):
   element = int(input())
   set2.add(element)

# Find the common elements and sort them in ascending order
common_elements = sorted(set1.intersection(set2))
print("Common elements in ascending order:", common_elements)
```

2) **Problem Statement:** Write a Python program that takes a string as input and counts the number of punctuation marks (e.g., commas, periods, exclamation marks, question marks) in the given text.

Solution:

```
test_str = str(input())
punc = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
cnt=0
for ele in test_str:
   if ele in punc:
      cnt+=1
print(cnt)
```

3) **Problem Statement:** Given an unsorted list of some elements, find the frequency of every element in the list using a Python dictionary.

Solution:

```
mylist = []
n=int(input())
for k in range(n):
   val = input()
   mylist.append(val)

freq = {}
for item in mylist:
   if (item in freq):
      freq[item] += 1
   else:
      freq[item] = 1

for key, value in freq.items():
   print(key,value,sep='-')
```

Bonus Question:

4) **Problem Statement:** Create a python program that translates a sentence into Pig Latin. In Pig Latin, words are transformed by moving the first letter to the end and adding "ay." For example, "hello" becomes "ellohay."

Solution:

```
sentence = input()

# Translate the sentence to Pig Latin
words = sentence.split()
pigwords=[]
for word in words:
   if word[0].isupper():
      pigword = word[1].capitalize() + word[2:] + word[0].lower() + "ay"
      pigwords.append(pigword)
   else:
      pigword = word[1:] + word[0] + "ay"
      pigwords.append(pigword)

pigsentence = ' '.join(pigwords)
print("Translated sentence:", pigsentence)
```

20-10-2023 (Friday):

Sections - K, W (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program that takes a sentence from the user and switches the first and last letter of every word in the sentence. Print the string thus obtained with all letters in uppercase.

Solution:

```
s = input()
words = s.split()
swapped = []

for i in words:
    if len(i) == 1:
        swapped.append(i)
    else:
        swapped.append(i[-1] + i[1:-1] + i[0])
s = " ".join(swapped)

print(s.upper())
```

2) **Problem Statement:** Write a program that takes two words as input from the user and displays the set of all unique letters that are present in one word but not the other. Treat upper and lower case characters as the same ie. T and t should be counted as a single entity etc. Display the characters in lowercase in ascending order.

Solution:

```
a = set(input().lower())
b = set(input().lower())
not_common = sorted(a.symmetric_difference(b))

if not_common:
    for letter in not_common:
        print(letter)
else:
    print(None)
```

3) **Problem Statement:** Write a program that takes a sentence as input, converts it to lower case and returns a dictionary where each key is a word from the sentence and its corresponding value is the number of vowels in the word.

Solution:

```
s = input()
# Converts string to lowercase before splitting
words = s.lower().split()
vowels = 'aeiou'
result = {}

for word in words:
   cnt = 0
   for letter in word:
      if letter in vowels:
         cnt += 1
   result[word] = cnt

print(result)
```

Bonus Question:

4) **Problem Statement:** A cipher is a method of transforming a message to conceal its meaning. The simplest technique involves shifting the letters in the message by a certain number of positions, known as the "shift" or "key".
Given a message and an integer shift value, print the encrypted message.
eg. Encryption - If shift value is 2, A becomes C, B becomes D etc. Z cycles back and becomes B.

Solution:

```
message = input()
shift = int(input())
result = ""

for i in range(len(message)):
   char = message[i]
   if char.isupper():
      result += chr((ord(char) + shift - 65) % 26 + 65)
   elif char.islower():
```

```
    result += chr((ord(char) + shift - 97) % 26 + 97)
  else:
    result += char

print(result)
```

20-10-2023 (Friday):

Sections - H, T (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program that takes a sentence from the user and converts the middle letter of every word in it to uppercase. If the string has even number of characters, convert both middle characters to upperase. Replace all whitespaces with hyphens (-). Display the new string thus obtained.

Solution:

```
s = input()
words = s.split()
middle = []
for i in words:
  m = len(i) // 2
  if len(i) % 2 == 1:
    middle.append(i[:m] + i[m].upper() + i[m+1:])
  else:
    middle.append(i[:m-1] + i[m-1:m+1].upper() + i[m+1:])
print("-".join(middle))
```

2) **Problem Statement:** Write a program that takes two words as input from the user and displays the set of all unique, common letters between them. Treat upper and lower case characters the same ie. T and t should be counted as a single entity etc. Display the characters in lower case in ascending order.

Solution:

```
a = set(input().lower())
b = set(input().lower())
common = sorted(a.intersection(b))
if common:
  for letter in common:
    print(letter)
```

```
else:
   print(None)
```

3) **Problem Statement:** Write a program that takes in a list of words as input, converts it to lower case and returns a dictionary where each key is a word from the input list and its corresponding value is the length of the word.

Solution:

```
n = int(input())
words = []
for i in range(n):
   w = input()
   words.append(w.lower())

word_len = {}
for i in words:
   word_len[i] = len(i)
print(word_len)
```

Bonus Question:

4) **Problem Statement:** A cipher is a method of transforming a message to conceal its meaning. The simplest technique involves shifting the letters in the message by a certain number of positions, known as the "shift" or "key".
Given an encrypted message and an integer shift value, print the decrypted message.
eg. Decryption - If shift value is 2, C becomes A, D becomes B etc. A cycles back and becomes Y.

Solution:

```
message = input()
shift = int(input())
result = ""
for i in range(len(message)):
   char = message[i]
   if char.isupper():
      result += chr((ord(char) - shift - 65) % 26 + 65)
   elif char.islower():
      result += chr((ord(char) - shift - 97) % 26 + 97)
```

```
    else:
        result += char
print(result)
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 6

Topic: Programs on Combination of Sets, Dictionaries and Strings

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Anchor and Lab Anchor – EC Campus: Prof. Kundhavai K R
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab session unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students to execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Input, Output, and Operators
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

Day: Monday
Sections: O2, C2, K2, B2, N2

Problem Statement 1:

Write a program that takes input of cricketer names as a string and finds the length of each name.

Solution:

```
cricketers_string = input("Enter the string of names(seperated by space): ")
cricketers = cricketers_string.split(' ')
item_lengths = [len(cricketer) for cricketer in cricketers]
for i, item in enumerate(cricketers):
    print(f"Length of {cricketers[i]}: {item_lengths[i]}")
```

Problem Statement 2:

Write a program that takes input of student SRN and marks in five subjects and creates a dictionary of SRN and total marks.
(SRN mark1 mark2 mark3 mark4 mark5)(separate the students using ,)

Solution:

```
students_string = input("Enter the list of students SRN and marks in five
subjects(SRN mark1 mark2 mark3 mark4 mark5)(seperate the students using ,):")
students_total_list = students_string.split(',')
students_dict = dict()
for i in students_total_list:
```

```
    student = i.split(' ')
    print(student)
    ans = 0
    for j in student[1:]:
        ans+=int(j)
    students_dict[student[0]] = ans
print(students_dict)
```

Problem Statement 3:

Write a program that takes information about cars in the specified format and create a dictionary with brand name as key value and list of unique CC values offered by the brand.
(BrandName,LaunchYear,ModelNumber,ModelName,CC,Price|)

Solution:

```
data_string = input("Enter the data in the following
format(BrandName,LaunchYear,ModelNumber,ModelName,CC,Price|): ")
car_entries = data_string.split('|')
car_info_dict = {}
for entry in car_entries:
    info = entry.split(',')
    brand = info[0]
    if brand not in car_info_dict:
        car_info_dict[brand] = set()
    car_info_dict[brand].add(info[4])
for brand, car_info_set in car_info_dict.items():
    print(f"Brand: {brand}")
    for info in car_info_set:
        print(f"Available CC: {info}")
```

Problem Statement 4:

Write a program that takes input of list of numbers as string and counts the number of occurrences of each element.

Solution:

```
numbers = input("Enter the numbers seperated by ,: ")
list_of_numbers = numbers.split(',')
```

```
freq = dict()
for item in list_of_numbers:
    if (item in freq):
        freq[item] += 1
    else:
        freq[item] = 1
print(freq)
```

Problem Statement 5:

Write a program to create a dictionary of students with their total marks and set of teachers given a string of students in the following format, (SRN,SubjectCode TeacherName marks,SubjectCode TeacherName marks)(Seperate students by ;)
Also calculate the total number of teachers who teach the students

Solution:

```
student_string = input("Enter the data in the following format(SRN,SubjectCode
TeacherName marks,SubjectCode TeacherName marks)(Seperate students by ;):")
students = student_string.split(';')
student_dict = dict()
teachers_count = 0
common_teachers = set()
for student in students:
    student_data = student.split(',')
    marks = 0
    teachers_set = set()
    for marks_data in student_data[1:]:
        data = marks_data.split(' ')
        marks+=int(data[2])
        teachers_set.add(data[1])
        common_teachers.add(data[1])
    teachers_list = ','.join(teachers_set)
    student_dict[student_data[0]] = (marks, teachers_list)
print(student_dict)
print("Total Number of teachers: "+str(len(common_teachers)))
```

Day: Tuesday
Sections: D2, I2, E2, L2

Problem Statement 1:

Write a program that takes input of footballers names as a string and finds the length of each name.

Solution:

```
footballers_string = input("Enter the string of footballer names (separated by
space): ")
footballers = footballers_string.split(' ')
item_lengths = [len(footballer) for footballer in footballers]
for i, footballer in enumerate(footballers):
    print(f"Length of {footballers[i]}: {item_lengths[i]}")
```

Problem Statement 2:

Write a program that takes input of employee ID and performance score in five projects and creates a dictionary of employee ID and total performance score.
(Emp ID pr1 pr2 pr3 pr4 pr5)(separate the employees using ,)

Solution:

```
employees_string = input("Enter the list of students employee ID and marks in
five projects(EMP ID pr1 pr2 pr3 pr4 pr5)(seperate the employees using ,):")
employees_total_list = employees_string.split(',')
employees_dict = dict()
for i in employees_total_list:
    employees = i.split(' ')
    print(employees)
    ans = 0
    for j in employees[1:]:
        ans+=int(j)
    employees_dict[employees[0]] = ans
print(employees_dict)
```

Problem Statement 3:

Write a program that takes information about bikes in the specified format and create a dictionary with brand name as key value and list of unique CC values offered by the brand.
(BrandName, LaunchYear, ModelNumber, ModelName, CC, Price|)

Solution:

```
data_string = input("Enter the data in the following
format(BrandName,LaunchYear,ModelNumber,ModelName,CC,Price|): ")
bike_entries = data_string.split('|')
bike_info_dict = {}
for entry in bike_entries:
    info = entry.split(',')
    brand = info[0]
    if brand not in bike_info_dict:
        bike_info_dict[brand] = set()
    bike_info_dict[brand].add(info[4])
for brand, bike_info_set in bike_info_dict.items():
    print(f"Brand: {brand}")
    for info in bike_info_set:
        print(f"Available CC: {info}")
```

Problem Statement 4:

Write a program that takes input of list of numbers as string and counts the number of occurrences of each element.

Solution:

```
numbers = input("Enter the numbers seperated by ,: ")
list_of_numbers = numbers.split(',')
freq = dict()
for item in list_of_numbers:
    if (item in freq):
        freq[item] += 1
    else:
        freq[item] = 1
```

```
print(freq)
```

Problem Statement 5:
Write a program to create a dictionary of athletes with their performance scores and set of coaches given a string of athletes in the following format, (AthleteID, Sport CoachName Score, Sport CoachName Score)(Separate athletes by ;). Also, calculate the total number of unique coaches.

Solution:

```
athlete_data = input("Enter athlete data in the specified format (AthleteID,
Sport CoachName Score;): ")
athlete_entries = athlete_data.split(';')
athlete_info_dict = {}
unique_coaches = set()

for athlete_entry in athlete_entries:
    info = athlete_entry.split(',')
    athlete_id = info[0].strip()
    sports_and_scores = info[1:]
    athlete_scores = []
    coach_set = set()

    for i in range(0, len(sports_and_scores), 3):
        sport = sports_and_scores[i].strip()
        coach_name = sports_and_scores[i + 1].strip()
        score = float(sports_and_scores[i + 2].strip())

        athlete_scores.append((sport, score))
        coach_set.add(coach_name)

    athlete_info_dict[athlete_id] = {"Scores": athlete_scores, "Coaches":
coach_set}
    unique_coaches.update(coach_set)

for athlete_id, info in athlete_info_dict.items():
    print(f"Athlete ID: {athlete_id}")
    print(f"Performance Scores: {', '.join([f'{sport}: {score}' for sport, score
in info['Scores']])}")
    print(f"Coaches: {', '.join(info['Coaches'])}")

print(f"Total Number of Unique Coaches: {len(unique_coaches)}")
```

Day: Wednesday
Sections: C1, K1, J2, F2

Problem Statement 1:

Write a program that takes input of prime ministers names as a string and finds the length of each name.

Solution:

```
prime_ministers_string = input("Enter the string of prime ministers' names
(separated by space): ")
prime_ministers = prime_ministers_string.split(' ')
item_lengths = [len(pm) for pm in prime_ministers]
for i, pm in enumerate(prime_ministers):
    print(f"Length of {prime_ministers[i]}: {item_lengths[i]}")
```

Problem Statement 2:

Write a program that takes input of product names and their sales figures in three different regions and creates a dictionary of product names and their total sales. The input should be in the format: (ProductName Region1Sales Region2Sales Region3Sales) with each product's data separated by commas. Calculate the total sales for each product and store it in the dictionary.

Solution:

```
product_sales_string = input("Enter the list of product names and sales figures
(ProductName Region1Sales Region2Sales Region3Sales)(separate the products using
,): ")
product_sales_list = product_sales_string.split(',')

product_sales_dict = {}

for item in product_sales_list:
    product_data = item.split(' ')
    product_name = product_data[0].strip()
    region1_sales = int(product_data[1].strip())
    region2_sales = int(product_data[2].strip())
    region3_sales = int(product_data[3].strip())
```

```
    total_sales = region1_sales + region2_sales + region3_sales

    product_sales_dict[product_name] = total_sales

print(product_sales_dict)
```

Problem Statement 3:

Write a program that takes information about laptops in the specified format and creates a dictionary with the manufacturer's name as the key and a list of unique processor types offered by the manufacturer. (ManufacturerName, LaunchYear, ModelNumber, ModelName, ProcessorType, Price|)

Solution:

```
data_string = input("Enter the data in the following format
(ManufacturerName,LaunchYear,ModelNumber,ModelName,ProcessorType,Price|): ")
laptop_entries = data_string.split('|')
laptop_info_dict = {}

for entry in laptop_entries:
    info = entry.split(',')
    manufacturer = info[0]
    if manufacturer not in laptop_info_dict:
        laptop_info_dict[manufacturer] = set()
    laptop_info_dict[manufacturer].add(info[4])

for manufacturer, processor_types in laptop_info_dict.items():
    print(f"Manufacturer: {manufacturer}")
    for processor_type in processor_types:
        print(f"Available Processor Types: {processor_type}")
```

Problem Statement 4:

Write a program that takes input of a list of words as a string and counts the number of occurrences of each word.

Solution:

```
words_input = input("Enter a list of words separated by commas: ")
word_list = words_input.split(',')

word_count = {}

for word in word_list:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1

print(word_count)
```

Problem Statement 5:

Write a program to create a dictionary of musicians with their album ratings and set of producers given a string of musicians in the following format, (MusicianID, Album ProducerName Rating, Album ProducerName Rating)(Separate musicians by ;). Also, calculate the total number of unique producers

Solution:

```
musician_data = input("Enter musician data in the specified format (MusicianID,
Album ProducerName Rating;): ")
musician_entries = musician_data.split(';')
musician_info_dict = {}
unique_producers = set()

for musician_entry in musician_entries:
    info = musician_entry.split(',')
    musician_id = info[0].strip()
    albums_and_ratings = info[1:]
    album_ratings = []
    producer_set = set()

    for i in range(0, len(albums_and_ratings), 3):
        album = albums_and_ratings[i].strip()
```

```
        producer_name = albums_and_ratings[i + 1].strip()
        rating = float(albums_and_ratings[i + 2].strip())

        album_ratings.append((album, rating))
        producer_set.add(producer_name)

    musician_info_dict[musician_id] = {"AlbumRatings": album_ratings,
"Producers": producer_set}
    unique_producers.update(producer_set)

for musician_id, info in musician_info_dict.items():
    print(f"Musician ID: {musician_id}")
    print(f"Album Ratings: {', '.join([f'{album}: {rating}' for album, rating in
info['AlbumRatings']])}")
    print(f"Producers: {', '.join(info['Producers'])}")

print(f"Total Number of Unique Producers: {len(unique_producers)}")
```

Day: Thursday
Sections: B1, N1, O1, D1, M1

Problem Statement 1:

Write a program that takes input of national heritages names as a string and finds the length of each name.

Solution:

```
heritage_names_string = input("Enter the string of national heritage names
(separated by space): ")
heritage_names = heritage_names_string.split(' ')
name_lengths = [len(name) for name in heritage_names]
for i, name in enumerate(heritage_names):
    print(f"Length of {heritage_names[i]}: {name_lengths[i]}")
```

Problem Statement 2:

Write a program that takes input of car models and their monthly sales figures in two different cities and creates a dictionary of car models and their total sales. The input should be in the format: (CarModel City1Sales City2Sales) with each car model's data separated by commas. Calculate the total sales for each car model and store it in the dictionary.

Solution:

```
data_string = input("Enter the data in the following format (CarModel,
City1Sales, City2Sales|): ")
car_entries = data_string.split('|')
car_sales_dict = {}

for entry in car_entries:
    info = entry.split(',')
    car_model = info[0]
    city1_sales = int(info[1])
    city2_sales = int(info[2])

    total_sales = city1_sales + city2_sales

    car_sales_dict[car_model] = total_sales
```

```
for car_model, total_sales in car_sales_dict.items():
    print(f"Car Model: {car_model}")
    print(f"Total Sales: {total_sales}")
```

Problem Statement 3:

Write a program that takes information about smartphones in the specified format and creates a dictionary with the manufacturer's name as the key and a list of unique screen sizes offered by the manufacturer. (ManufacturerName, LaunchYear, ModelNumber, ModelName, ScreenSize, Price|)

Solution:

```
data_string = input("Enter the data in the following format (ManufacturerName,
LaunchYear, ModelNumber, ModelName, ScreenSize, Price|): ")
smartphone_entries = data_string.split('|')
smartphone_info_dict = {}

for entry in smartphone_entries:
    info = entry.split(',')
    manufacturer = info[0]
    if manufacturer not in smartphone_info_dict:
        smartphone_info_dict[manufacturer] = set()
    smartphone_info_dict[manufacturer].add(info[4])

for manufacturer, screen_sizes in smartphone_info_dict.items():
    print(f"Manufacturer: {manufacturer}")
    print(f"Unique Screen Sizes: {', '.join(screen_sizes)}")
```

Problem Statement 4:

Write a program that takes input of a list of cities as a string and counts the number of occurrences of each city.

Solution:

```
cities_input = input("Enter a list of cities separated by commas: ")
city_list = cities_input.split(',')

city_count = {}
```

```
for city in city_list:
    if city in city_count:
        city_count[city] += 1
    else:
        city_count[city] = 1

print(city_count)
```

Problem Statement 5:

Write a program to create a dictionary of software applications with their user ratings and a set of developers given a string of applications in the following format, (AppID, Developer DeveloperName Rating, Developer DeveloperName Rating)(Separate applications by ;). Also, calculate the total number of unique developers.

Solution:

```
app_data = input("Enter software application data in the specified format (AppID, Developer DeveloperName Rating, Developer DeveloperName Rating;): ")
app_entries = app_data.split(';')
app_info_dict = {}
unique_developers = set()

for app_entry in app_entries:
    info = app_entry.split(',')
    app_id = info[0].strip()
    developers_and_ratings = info[1:]
    developer_ratings = []
    developer_set = set()

    for i in range(0, len(developers_and_ratings), 3):
        developer = developers_and_ratings[i].strip()
        developer_name = developers_and_ratings[i + 1].strip()
        rating = float(developers_and_ratings[i + 2].strip())

        developer_ratings.append((developer, rating))
        developer_set.add(developer_name)

    app_info_dict[app_id] = {"DeveloperRatings": developer_ratings, "Developers": developer_set}
```

```
    unique_developers.update(developer_set)

for app_id, info in app_info_dict.items():
    print(f"App ID: {app_id}")
    print(f"User Ratings: {', '.join([f'{developer}: {rating}' for developer,
rating in info['DeveloperRatings']])}")
    print(f"Developers: {', '.join(info['Developers'])}")

print(f"Total Number of Unique Developers: {len(unique_developers)}")
```

Day: Friday
Sections: E1, L1, J1, F1, A1, I1

Problem Statement 1:

Write a program that takes input of satellites names as a string and finds the length of each name.

Solution:

```
satellite_names_string = input("Enter the string of satellite names (separated by
space): ")
satellite_names = satellite_names_string.split(' ')
name_lengths = [len(name) for name in satellite_names]
for i, name in enumerate(satellite_names):
    print(f"Length of {satellite_names[i]}: {name_lengths[i]}")
```

Problem Statement 2:

Write a program that takes input of restaurant names and their monthly revenue figures in two different locations and creates a dictionary of restaurant names and their total revenue. The input should be in the format: (RestaurantName Location1Revenue Location2Revenue) with each restaurant's data separated by commas. Calculate the total revenue for each restaurant and store it in the dictionary.

Solution:

```
data_string = input("Enter the data in the following format (ProductName,
Region1Sales, Region2Sales, Region3Sales|): ")
product_entries = data_string.split('|')
product_sales_dict = {}

for entry in product_entries:
    info = entry.split(',')
    product_name = info[0]
    region1_sales = int(info[1])
    region2_sales = int(info[2])
    region3_sales = int(info[3])
```

```
    total_sales = region1_sales + region2_sales + region3_sales

    product_sales_dict[product_name] = total_sales

for product_name, total_sales in product_sales_dict.items():
    print(f"Product Name: {product_name}")
    print(f"Total Sales: {total_sales}")
```

Problem Statement 3:

Write a program that takes information about clothing brands in the specified format and creates a dictionary with the brand name as the key and a list of unique clothing sizes offered by the brand. (BrandName, LaunchYear, ModelNumber, ProductName, Size, Price|)

Solution:

```
data_string = input("Enter the data in the following format (BrandName,
LaunchYear, ModelNumber, ProductName, Size, Price|): ")
clothing_entries = data_string.split('|')
clothing_info_dict = {}

for entry in clothing_entries:
    info = entry.split(',')
    brand_name = info[0]
    if brand_name not in clothing_info_dict:
        clothing_info_dict[brand_name] = set()
    clothing_info_dict[brand_name].add(info[4])

for brand_name, sizes in clothing_info_dict.items():
    print(f"Brand Name: {brand_name}")
    print(f"Unique Clothing Sizes: {', '.join(sizes)}")
```

Problem Statement 4:

Write a program that takes input of a list of book genres as a string and counts the number of occurrences of each book genre.

Solution:

```
brands_input = input("Enter a list of car brand names separated by commas: ")
brand_list = brands_input.split(',')

brand_count = {}

for brand in brand_list:
    if brand in brand_count:
        brand_count[brand] += 1
    else:
        brand_count[brand] = 1

print(brand_count)
```

Problem Statement 5:

Write a program to create a dictionary of restaurants with their customer ratings and set of chefs given a string of restaurants in the following format, (RestaurantID, Chef ChefName Rating, Chef ChefName Rating)(Separate restaurants by ;). Also, calculate the total number of unique chefs.

Solution:

```
restaurant_data = input("Enter restaurant data in the specified format
(RestaurantID, Chef ChefName Rating, Chef ChefName Rating;): ")
restaurant_entries = restaurant_data.split(';')
restaurant_info_dict = {}
unique_chefs = set()

for restaurant_entry in restaurant_entries:
    info = restaurant_entry.split(',')
    restaurant_id = info[0].strip()
    chefs_and_ratings = info[1:]
    chef_ratings = []
    chef_set = set()

    for i in range(0, len(chefs_and_ratings), 3):
        chef = chefs_and_ratings[i].strip()
        chef_name = chefs_and_ratings[i + 1].strip()
        rating = float(chefs_and_ratings[i + 2].strip())

        chef_ratings.append((chef, rating))
        chef_set.add(chef_name)
```

```
    restaurant_info_dict[restaurant_id] = {"ChefRatings": chef_ratings, "Chefs":
chef_set}
    unique_chefs.update(chef_set)

for restaurant_id, info in restaurant_info_dict.items():
    print(f"Restaurant ID: {restaurant_id}")
    print(f"Customer Ratings: {', '.join([f'{chef}: {rating}' for chef, rating in
info['ChefRatings']])}")
    print(f"Chefs: {', '.join(info['Chefs'])}")

print(f"Total Number of Unique Chefs: {len(unique_chefs)}")
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 10

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Programs on Recursion and callbacks
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

20-11-2023 (Monday):
Sections - L,X (08:00 AM to 9:30 AM):
(Combined with Lab 9)

20-11-2023 (Monday):
Sections - C, O (11:30 AM to 1:00 PM):

1) **Problem Statement:** Complete the recursive function decimal_to_binary(n) that takes an integer n as input and returns its binary representation as a string.
Solution:

```
def decimal_to_binary(n):
    if n == 0:
        return '0'
    elif n == 1:
        return '1'
    else:
        return decimal_to_binary(n // 2) + str(n % 2)
```

2) **Problem Statement:** Complete the function, is_power_of_4(n), using recursion that takes a positive integer n as input and returns 'T' if n is a power of 4, and 'F' otherwise.
Solution:

```
def is_power_of_4(n):
    if n == 1:
        return 'T'
    elif n < 1 or n % 4 != 0:
```

```
    return 'F'
  else:
    return is_power_of_4(n // 4)
```

3) **Problem Statement:** A child is running up a staircase with n steps and can hop either 1, 2, or 3 steps at a time. Complete the recursive function, count_ways_to_reach(n), that counts the number of distinct ways the child can reach the top of the staircase.

Solution:

```
def count_ways_to_reach(n):
  if n == 0:
    return 1
  elif n < 0:
    return 0
  else:
        return count_ways_to_reach(n - 1) + count_ways_to_reach(n - 2) + count_ways_to_reach(n - 3)
```

Bonus Question:

1) **Problem Statement:** You are given two integers 'm' and 'n'. A grid of size m x n can be formed using it. Write a recursive function, unique_paths(m, n), to find the number of unique paths from the top-left corner to the bottom-right corner. You can only move down or right. You don't have to form a grid, you can find the solution by just using the integers 'm' and 'n'.

Solution:

```
def unique_paths(m, n):
  # Base case: when we reach the bottom-right corner, there is one unique path
  if m == 1 and n == 1:
    return 1

  # Recursive cases: move either down or right and sum the unique paths
  paths_down = unique_paths(m - 1, n) if m > 1 else 0
  paths_right = unique_paths(m, n - 1) if n > 1 else 0

  return paths_down + paths_right
```

20-11-2023 (Monday):
Sections - B, N (01:45 AM to 3:15 PM);

1) **Problem Statement:** Complete the recursive function is_power_of_2(n) that checks whether a given positive integer n is a power of 2. The function should return 'T' if n is a power of 2 and 'F' otherwise.
Solution:

```
def is_power_of_2(n):
    if n == 1:
        return 'T'
    elif n < 1 or n % 2 != 0:
        return 'F'
    else:
        return is_power_of_2(n // 2)
```

2) **Problem Statement:** A child is running up a staircase with n steps and can hop either 1 or 2 steps at a time. Complete recursive function, count_ways_to_reach(n), that counts the number of distinct ways the child can reach the top of the staircase..
Solution:

```
def count_ways_to_reach(n):
    if n == 0:
        return 1
    elif n < 0:
        return 0
    else:
        return count_ways_to_reach(n - 1) + count_ways_to_reach(n - 2)
```

3) **Problem Statement:** Complete the recursive function decimal_to_hexadecimal(n) that takes a positive decimal number 'n' as input and returns its hexadecimal representation as a string. The function should use the provided hex_digits string, which contains the hexadecimal digits "0123456789ABCDEF".
Solution:

```
def decimal_to_hexadecimal(n):
    hex_digits = "0123456789ABCDEF"
    if n < 16:
```

```
    return hex_digits[n]
  else:
    return decimal_to_hexadecimal(n // 16) + hex_digits[n % 16]
```

Bonus Question:

1) **Problem Statement:** Complete the recursive function is_interleave(s1, s2, s3) that takes three strings, s1, s2, and s3, as input and determines whether s3 is formed by interleaving the characters of s1 and s2. Interleaving here means maintaining the relative ordering of characters from both strings. The function should return 'T' if the interleaving is possible, and 'F' otherwise.

   Solution:

```
def is_interleave(s1, s2, s3):
  # Check if all strings are empty, indicating a successful interleaving
  if not s1 and not s2 and not s3:
    return 'T'

  # Check if the first character of s1 matches s3, and recurse with updated strings
  if s1 and s3 and s1[0] == s3[0] and is_interleave(s1[1:], s2, s3[1:]) == 'T':
    return 'T'

  # Check if the first character of s2 matches s3, and recurse with updated strings
  if s2 and s3 and s2[0] == s3[0] and is_interleave(s1, s2[1:], s3[1:]) == 'T':
    return 'T'

  # If no match is found, return 'F'
  return 'F'
```

21-11-2023 (Tuesday):

Sections - F, R (8:00AM to 10:30AM)

1) **Problem Statement:** Implement a recursive function to perform basic string compression. For example, "aabbbccc" would become "a2b3c3".

   Solution:

```
input_string = input()

def compress_string(s):
```

```
    if not s:
        return ""

    count = 1
    compressed = ""

    # Iterate through the string and count consecutive occurrences
    while len(s) > 1 and s[0] == s[1]:
        count += 1
        s = s[1:]

    # Append the character and its count to the compressed string
    compressed += s[0] + str(count)

    # Recursively compress the remaining string
    compressed += compress_string(s[1:])

    return compressed


result = compress_string(input_string)
print(result)
```

2) **Problem Statement:** Create a recursive function to remove consecutive duplicate characters in a string.

Solution:

```
input_string = input()

def remove_consecutive_duplicates(s):
    if len(s) <= 1:
        return s

    if s[0] == s[1]:
        # If consecutive characters are the same, skip the first character
        return remove_consecutive_duplicates(s[1:])
    else:
        # If consecutive characters are different, keep the first character
        return s[0] + remove_consecutive_duplicates(s[1:])
```

```
result = remove_consecutive_duplicates(input_string)
print(result)
```

3) **Problem Statement:** Write a code using recursion to find the GCD(Greatest Common Divisor) of 2 given numbers

Solution:

```
a = int(input())
b = int(input())

def gcd(a,b):

   if (a == 0):
      return b
   if (b == 0):
      return a

   # base case of recursion
   if (a == b):
      return a

   # when a is greater
   if (a > b):
      return gcd(a-b, b)
   # when b is greater
   return gcd(a, b-a)

result = gcd(a,b)
print(result)
```

4) **(BONUS) Problem Statement:** Write a program in python using recursion to find all the permutations of a string in lexicographical order

Solution:

```
input_string = input()
```

```
def get_permutations(s):
    if len(s) <= 1:
        return [s]

    # Recursive step: get permutations of substring excluding the first character
    smaller_permutations = get_permutations(s[1:])

    char = s[0]
    permutations = []

    # Insert the first character at all possible positions in each smaller permutation
    for smaller_permutation in smaller_permutations:
        for i in range(len(smaller_permutation) + 1):
            new_permutation = smaller_permutation[:i] + char + smaller_permutation[i:]
            permutations.append(new_permutation)

    return permutations

permutations_result = get_permutations(input_string)
for perm in sorted(permutations_result):
    print(perm)
```

21-11-2023 (Tuesday):

Sections - A, M (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program using recursion, to check if a given string is a palindrome in python

   Solution:

```
input_string = input()

def is_palindrome(s):
    # Base case: an empty string or a string with one character is a palindrome
    if len(s) <= 1:
        return True
    # Check if the first and last characters are the same
    elif s[0] == s[-1]:
        # Recursively check the substring without the first and last characters
```

```
    return is_palindrome(s[1:-1])
  else:
    # If the first and last characters are different, it's not a palindrome
    return False
result = is_palindrome(input_string)
print(result)
```

2) **Problem Statement:** Calculate the Sum of Even Numbers in a string of numbers using recursive functions

Solution:

```
def sum_of_even_digits(number, position=0):
  num_str = str(number)

  # Base case: If the position is out of bounds, return 0
  if position >= len(num_str):
    return 0

  # Recursive case: Add the digit at the even position and move to the position
  else:
    digit = int(num_str[position])
    if digit % 2 == 0:  # Check if the digit is even
      return digit + sum_of_even_digits(number, position + 1)
    else:
      return sum_of_even_digits(number, position + 1)

# Example usage:
user_input = input("Enter an integer: ")
result = sum_of_even_digits(user_input)
print(result)
```

3) **Problem Statement:** Calculate the Sum of Digits at Even Positions using recursive functions

Solution:

```
def sum_of_digits_at_even_positions(user_input, position=0):
  # Base case: If the position is out of bounds, return 0
  if position >= len(user_input):
    return 0
```

```
    # Recursive case: Add the digit at the even position and move to the next even position
    else:
        digit = int(user_input[position])
        if position % 2 == 0:  # Check if the position is even
            return digit + sum_of_digits_at_even_positions(user_input, position + 2)
        else:
            return sum_of_digits_at_even_positions(user_input, position + 2)

# Example usage:
user_input = input("Enter a string of digits: ")
result = sum_of_digits_at_even_positions(user_input)
print(f"The sum of digits at even positions in '{user_input}' is: {result}")
```

4) **(BONUS) Problem Statement:** Write a program in python to find the length of the longest common subsequence of two strings.

Solution:

```
string1 = input()
string2 = input()

def longest_common(str1, str2):
    # Base case: if either of the strings is empty, there is no common subsequence
    if not str1 or not str2:
        return 0

    # Case 1: Last characters of both strings match
    if str1[-1] == str2[-1]:
        return 1 + longest_common(str1[:-1], str2[:-1])

    # Case 2: Last characters don't match, consider two possibilities (exclude one character at a time)
    else:
        option1 = longest_common(str1, str2[:-1])
        option2 = longest_common(str1[:-1], str2)
        return max(option1, option2)



result = longest_common(string1, string2)
print(result)
```

21-11-2023 (Tuesday):
Sections - D, P (1:45PM to 3:15PM)

1) **Problem Statement:** You are given a string s. Write a program to create a recursive function to remove consecutive duplicate characters in a string.

Solution:

```
input_string = input()

def remove_consecutive_duplicates(s):
    if len(s) <= 1:
        return s

    if s[0] == s[1]:
        # If consecutive characters are the same, skip the first character
        return remove_consecutive_duplicates(s[1:])
    else:
        # If consecutive characters are different, keep the first character
        return s[0] + remove_consecutive_duplicates(s[1:])


result = remove_consecutive_duplicates(input_string)
print(result)
```

2) **Problem Statement:** You are given 2 numbers. Write a code using recursion to find the GCD(Greatest Common Divisor) of 2 given numbers

Solution:

```
a = int(input())
b = int(input())

def gcd(a,b):

    if (a == 0):
        return b
    if (b == 0):
        return a

    # base case of recursion
    if (a == b):
        return a
```

```
    # when a is greater
    if (a > b):
        return gcd(a-b, b)
    # when b is greater
    return gcd(a, b-a)

result = gcd(a,b)
print(result)
```

3) **Problem Statement: Problem Statement:** Write a code using recursion to find the sum of values in the row of pascal's triangle given the row number. Each number in Pascal's Triangle is the sum of the two numbers directly above it in the previous row. The first and last numbers in each row are always 1, as they don't have two numbers above them.

Solution:

```
num = int(input())

def pascal(n):
    if n == 1:
        return [1]
    else:
        line = [1]
        previous_line = pascal(n-1)
        for i in range(len(previous_line)-1):
            line.append(previous_line[i] + previous_line[i+1])
        line += [1]
    return line

l = pascal(num)
x = sum(l)
print(x)
```

4) **(BONUS) Problem Statement:** You are given a string. Write a program in python using recursion to find all the permutations of a string in lexicographical order

Solution:

```
input_string = input()
```

```
def get_permutations(s):
    if len(s) <= 1:
        return [s]

    # Recursive step: get permutations of substring excluding the first character
    smaller_permutations = get_permutations(s[1:])

    char = s[0]
    permutations = []

    # Insert the first character at all possible positions in each smaller permutation
    for smaller_permutation in smaller_permutations:
        for i in range(len(smaller_permutation) + 1):
            new_permutation = smaller_permutation[:i] + char + smaller_permutation[i:]
            permutations.append(new_permutation)

    return permutations

permutations_result = get_permutations(input_string)
for perm in sorted(permutations_result):
    print(perm)
```

22-11-2023 (Wednesday):
Sections - Y (1:45PM to 3:15PM)
(Combined with Lab 9)

23-11-2023 (Thursday):
Sections - I, U (8:00AM to 9:30 AM)

1) **Problem Statement:** Write a program to check if the given number is a power of 3 using recursion.

   Solution:

```
def power(n):
```

```
    if n==1:
        return True
    elif int(n)!=n or n<=0:
        return False
    else:
        return power(n/3)
n=int(input())
print(power(n))
```

2) **Problem Statement:** Write a program to get the count of the special characters in a given string using recursion.

Solution:

```
def get_special(s):
    if not s:
        return 0
    elif not s[0].isalnum() and not s[0].isspace():
        return 1 + get_special(s[1:])
    else:
        return get_special(s[1:])
s=input()
print(get_special(s))
```

3) **Problem Statement:** Write a program to find the largest number in a given list, using recursion.

Solution:

```
def find_largest(nums):
    if not nums:
        return None
    if len(nums) == 1:
        return nums[0]
    mymax = find_largest(nums[1:])
    return nums[0] if nums[0] > mymax else mymax
n=int(input())
```

```
s = input()
numbers = list(map(int, s.split()))
print(find_largest(numbers))
```

Bonus Question:

4) **Problem Statement:** Write a program to convert a given non-negative integer into its word representation. The word representation should be in the form of ones,tens,hundereds and thousands. For example, the integer 1234 should be represented as "1 thousands 2 hundreds 3 tens 4 ones"

Solution:

```
d={0:"zero",1:"one",2:"two",3:"three",4:"four",5:"five",6:"six",7:"seven",8:"eight",9:"ni
ne"}
places={1:"ones",2:"tens",3:"hundreds",4:"thousands"}
def int_to_str(num,string,cnt):
    if cnt==0:
        return string
    else:
        string+=d[int(num[-1])]+" "+places[cnt]+" "
        return int_to_str(num[:len(num)-1],string,cnt-1)
number=int(input())
number=str(number)[::-1]
print(int_to_str(number,"",len(number)))
```

23-11-2023 (Thursday):
Sections - E, Q (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program to convert a binary number represented as a string to its decimal equivalent, using recursion.

Solution:

```
def binary_to_decimal(s):
    if not s:
        return 0
    else:
```

```
    x=int(s[0]) * (2 ** (len(s) - 1))
    return  x + binary_to_decimal(s[1:])
binary_number = input()
print( binary_to_decimal(binary_number))
```

2) **Problem Statement:** Write a program to find the smallest number in a given list, using recursion.

Solution:

```
def find_smallest(nums):
  if not nums:
    return None
  if len(nums) == 1:
    return nums[0]
  mymax = find_smallest(nums[1:])
  return nums[0] if nums[0] < mymax else mymax
n=int(input())
s = input()
numbers = list(map(int, s.split()))
print(find_smallest(numbers))
```

3) **Problem Statement:** Write a program to count the number of occurences of a character in a given string, using recursion.

Solution:

```
def count_of_occurrences(s, char):
  if not s:
    return 0
  elif s[0] == char:
    return 1 + count_of_occurrences(s[1:], char)
  else:
    return count_of_occurrences(s[1:], char)

s = input()
character = input()
print(count_of_occurrences(s, character))
```

Bonus Question:

4) **Problem Statement:** Write a recursive function to remove consecutive repeated characters from a string.

Solution:

```
def remove_repeated(s):
    if len(s) <= 1:
        return s
    elif s[0] == s[1]:
        return remove_repeated(s[1:])
    else:
        return s[0] + remove_repeated(s[1:])

input_string = input()
print( remove_repeated(input_string))
```

23-11-2023 (Thursday):
Sections - G, S (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program using recursion, to check if a given string is a palindrome in python.

Solution:

```
input_string = input()

def is_palindrome(s):
    # Base case: an empty string or a string with one character is a palindrome
    if len(s) <= 1:
        return True
    # Check if the first and last characters are the same
    elif s[0] == s[-1]:
        # Recursively check the substring without the first and last characters
        return is_palindrome(s[1:-1])
    else:
        # If the first and last characters are different, it's not a palindrome
        return False
result = is_palindrome(input_string)
print(result)
```

2) **Problem Statement:** Write a recursive function to count the number of words in a given string. Implement a function word_count(s) that takes a string s as input and returns the count of words in the string using recursion.

Solution:

```
def word_count(s):
  if not s.strip():  # Check if the string is empty or contains only whitespace
    return 0
  else:
    # Split the string by spaces to get individual words
    words = s.split(" ", 1)

    # If there's only one word left, return 1
    if len(words) == 1:
      return 1
    else:
      # Recursively count the words in the rest of the string after first word
      return 1 + word_count(words[1])
```

3) **Problem Statement:** Write python recursive function to generate permutations of a given set of elements.

Solution:

```
def permutations(set):
 if len(set) == 0:
  return [[]]
 else:
  result = []
  for element in set:
   for permutation in permutations(set - {element}):
    result.append([element] + permutation)
  return result
```

Bonus Question:

4) **Problem Statement:** Given n disks initially stacked in ascending order of size on one rod (source rod), along with two additional rods (auxiliary and destination), the objective is to transfer the entire stack of disks from the source rod to the

destination rod, using the auxiliary rod for intermediate storage, following the rules mentioned above.
Develop a recursive algorithm in Python to solve the Tower of Hanoi problem for n disks, providing the sequence of moves required to achieve the transfer from the source rod to the destination rod.

Solution:

```
def tower_of_hanoi(n, source, destination, auxiliary):
    if n == 1:
        print(f"Move disk 1 from rod {source} to rod {destination}")
        return

    tower_of_hanoi(n - 1, source, auxiliary, destination)
    print(f"Move disk {n} from rod {source} to rod {destination}")
    tower_of_hanoi(n - 1, auxiliary, destination, source)

# Example usage
num_disks = int(input('num_disks:'))
source = input('source:')
destination = input('destination:')
auxiliary = input('auxiliary:')
tower_of_hanoi(num_disks, source, destination, auxiliary)
```

24-11-2023 (Friday):
Sections - J, V (8:00AM to 9:30 AM)

1) **Problem Statement:** Write a recursive function to decompress a string. The input string will have characters followed by a number indicating how many times that character should appear consecutively. For example, "a2b3c3" would become "aabbbccc".

Solution:

```
def decompress_string(s):
    if not s:
        return ""

    def decompress_helper(s, index):
        if index >= len(s):
            return ""
```

```
    char = s[index]
    count = 0
    index += 1

    while index < len(s) and s[index].isdigit():
      count = count * 10 + int(s[index])
      index += 1

    return char * count + decompress_helper(s, index)

  return decompress_helper(s, 0)
```

2) **Problem Statement:** Write a python code using recursion to find the lowest common multiple of 2 given numbers.

Solution:

```
# Function to calculate GCD using recursion (Euclidean algorithm)
def gcd(a, b):
  if b == 0:
    return a
  else:
    return gcd(b, a % b)

# Function to calculate LCM using recursion
def lcm(a, b):
  return (a * b) // gcd(a, b)
```

3) **Problem Statement:** Given a set of non-negative integers and a target sum, determine if there's a subset that sums up to the target. Solve this problem using python recursion.

Solution:

```
def isSubsetSum(nums, target):
  # Base cases
  if target == 0:
    return True
  if not nums or target < 0:
    return False
```

```
    include_current = isSubsetSum(nums[:-1], target - nums[-1])

    # Check if the current element is excluded from the subset
    exclude_current = isSubsetSum(nums[:-1], target)
    return include_current or exclude_current

lst = []
n = int(input("Enter n:"))
for i in range(0, n):
    ele = int(input())
    lst.append(ele)
target = int(input('target:'))
print(isSubsetSum(lst, target))
```

Bonus Question:

4) **Problem Statement:** Given an integer array of coins[ ] of size N representing different types of denominations and an integer sum, the task is to count the number of coins required to make a given value sum.

Solution:

```
def count(coins, n, sum):
    # If sum is 0 then there is 1 solution
    if (sum == 0):
        return 1

    # If sum is less than 0 then no solution exists
    if (sum < 0):
        return 0

    # If there are no coins and sum
    # is greater than 0, then no
    # solution exist
    if (n <= 0):
        return 0

    return count(coins, n - 1, sum) + count(coins, n, sum-coins[n-1])

coins = []
```

```
n = int(input("Enter n:"))
for i in range(0, n):
   ele = int(input())
   coins.append(ele)
sum = int(input('enter sum:'))
print(count(coins, n, sum))
```

24-11-2023 (Friday):
Sections - K, W (11:30AM to 1:00PM)

1) **Problem Statement:** The octal number system is a base-8 numbering system that uses digits from 0 to 7 to represent numbers. In this system, each digit represents a power of 8, starting from the rightmost digit. Write a recursive program to convert an octal number to decimal.

   Solution:

```
def octal_to_decimal(o):
   if o == 0:
      return 0
   else:
      return (o % 10) + (8 * octal_to_decimal(int(o / 10)))
```

2) **Problem Statement:** Write a program to find the sum of digits of a positive number without using loops.

   Solution:

```
def sum_of_digits(n):
   if n == 0:
      return 0
   else:
      return n % 10 + sum_of_digits(n // 10)
```

3) **Problem Statement:** Write a recursive function to find the number of vowels in a string.

Solution:

```
vowels = "aeiou"
def count_vowel(s):
    if not s:
        return 0
    elif s[0].lower() in vowels:
        return 1 + count_vowel(s[1:])
    else:
        return count_vowel(s[1:])
```

Bonus Question:

4) **Problem Statement:** Write a recursive function that takes a string and counts the number of substrings starting and ending with the same letter.
Eg. abca has four substrings satisfying the above condition - abca, a, b, c

Solution:

```
def count_sub(s, n):
    # Base case
    if not s:
        return 0

    # Subtracting to remove repeated cases
    res = count_sub(s[1:], n-1) + count_sub(s[:n-1], n-1) - count_sub(s[1:n-1], n-2)

    # Same characters
    if s[0] == s[-1]:
        res += 1
    return res
```

24-11-2023 (Friday):
Sections - H, T (1:45PM to 3:15PM)

1) **Problem Statement:** The octal number system is a base-8 numbering system that uses digits from 0 to 7 to represent numbers. In this system, each digit represents a power of 8, starting from the rightmost digit. Write a recursive program to convert a decimal number to octal.

Solution:

```
def decimal_to_octal(decimal):
    if decimal == 0:
        return 0
    else:
        return (decimal % 8) + (10 * decimal_to_octal(int(decimal / 8)))
```

2) **Problem Statement:** Write a program to remove all vowels from a string without the use of loops.

Solution:

```
vowels = "aeiou"
def no_vowel(s):
    if not s:
        return ""
    elif s[0].lower() in vowels:
        return "" + no_vowel(s[1:])
    else:
        return s[0] + no_vowel(s[1:])
```

3) **Problem Statement:** Write a program that takes a decimal number and finds the number of set bits in its binary representation.

Solution:

```
def dtob(n):
    if n == 0:
        return 0
    if n % 2 == 1:
        return 1 + dtob(n // 2)
    else:
        return dtob(n // 2)
```

Bonus Question:

4) **Problem Statement:** You are given two integers 'm' and 'n'. A grid of size m x n can be formed using it. Write a recursive function, unique_paths(m, n), to find the number of unique paths from the top-left corner to the bottom-right corner. You can

only move down or right. You don't have to form a grid, you can find the solution by just using the integers 'm' and 'n'.

Solution:

```
def unique_paths(m, n):
    # Base case: when we reach the bottom-right corner, there is one unique path
    if m == 1 and n == 1:
        return 1

    # Recursive cases: move either down or right and sum the unique paths
    paths_down = unique_paths(m - 1, n) if m > 1 else 0
    paths_right = unique_paths(m, n - 1) if n > 1 else 0

    return paths_down + paths_right
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week #10

Topic: Recursion and callbacks

Semester: I
Course Code: UE23CS151A Course
Anchor: Prof. Sindhu R Pai Lab
Anchor: Dr. Divyashree N
Anchor and Lab Anchor – EC Campus: Prof. Kundhavai K R
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab session unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students to execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Program questions on Input, Output, and Operators
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

Day: Monday
Sections: O2, C2, K2, B2, N2

Problem Statement 1: Write a recursive function to find the sum of all elements in a list of Integers. Take list of integers as user input.

Solution:

```
def rec_sum(arr):
    if len(arr) == 1:
        return arr[0]
    else:
        return arr[0] + rec_sum(arr[1:])
input_numbers = input("Enter a list of numbers separated by comma:")
number_list = input_numbers.split(",")
numbers = []
for number in number_list:
    numbers.append(int(number))
result = rec_sum(numbers)
print(result)
```

```
Enter a list of numbers separated by comma:10,20,30,40,50
150
```

Problem Statement 2: Write a function that takes a list of numbers and a callback function to find the sum of the numbers based on given conditions:

i. Sum of all even numbers in the list.

ii. Sum of all numbers not divisible by 3 in the list.

Solution:

```
def conditional_sum(numbers, callback):
    total = 0
    for num in numbers:
        if callback(num):
            total += num
    return total
def is_even(x):
    return x % 2 == 0
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = conditional_sum(numbers, is_even)
print(result)  # Output: 30 (since 2 + 4 + 6 + 8 + 10 = 30)
```

```
def conditional_sum(numbers, callback):
    total = 0
    for num in numbers:
        if callback(num):
            total += num
    return total
def not_divisible_by_3(x):
    return x % 3 != 0
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = conditional_sum(numbers, not_divisible_by_3)
print(result)  # Output: 54 (since 6 + 12 + 15 + 18 + 24 = 75)
```

Problem Statement 3: Write a python function that takes a list of strings as a callback function. It should apply the callback to each string in the list, but only return the results for strings that meet the following condition:

i) Find all the strings starting with Letter 'A'

ii) Find all the strings whose length is greater than 4.

Solution:

```python
def callback(strings, callback):
    results = []
    for string in strings:
        if callback(string):
            results.append(string)
    return results
def starts_with_a(string):
    return string.startswith('A')
def length_four(string):
    return len(string) > 4
strings = ['Antarctica', 'America', 'Boston', 'Canada', 'Africa', 'Pesu', 'Ben','Aero']
result = callback(strings, starts_with_a)
print(result)  #['Antarctica', 'America', 'Africa', 'Aero']
result = callback(strings, length_four)
print(result) #['Antarctica', 'America', 'Boston', 'Canada', 'Africa']
```

Problem Statement 4: Write a Python program that takes an integer 'n' as input and determines whether it is a power of 2. The program should return True if the input number is a power of 2 or False otherwise using recursion

Solution:

Solution:

```
def is_power_of_two(n):
  if n <= 0:
    return False
  elif n == 1:
    return True
  elif n % 2 == 0:
    return is_power_of_two(n // 2)
  else:
    return False

# Input from the user
num = int(input("Enter a number: "))
result = is_power_of_two(num)

if result:
  print("True")
else:
  print("False")
```

Sample Input and Output:

1) Sample Input 1: 16
   Sample Output 1: True
2) Sample Input 2: 18
   Sample Output 2: False

Day: Tuesday Sections:
D2, I2, E2, L2

Problem Statement 1: Write a Python program that takes an integer as input 'n' and counts the number of digits in the binary representation of 'n' using recursion.
Solution:

```
def count_binary_digits(n):
    if n <= 1:
        return 1
    else:
        return 1 + count_binary_digits(n // 2)
n = int(input("Enter a number: "))
result = count_binary_digits(n)
print(result)
```

```
Enter a number: 7
3
```

Problem Statement 2:

Write a recursive function to find the greatest common divisor (GCD) of two positive integers 'a' and 'b', using Euclidian algorithm.

1. Division Method
2. Subtraction Method

Solution: Division method

```
def gcd(a,b):
    if b == 0:
        return a
    else:
        return gcd(b,a%b)
print(gcd(20,12))
```

```
4
```

Solution: Subtraction method

```
#Euclidean algorithm by subtraction
def gcd(a,b):
    if a == b:
        res = a
    elif a>b:
        res = gcd(a-b,b)
    else:
        res = gcd(a,b-a)
    return res
print(gcd(24,87))
```

```
3
```

Problem Statement 3: Write a callback function to display the length (no of characters) in the files "sample.txt" and "test.txt".
Note: Create the files with some content.

```
def fun(l):
    print("The length of the file is:",l)
def filelen(path,a):
    f = open(path,"r")
    length = len(f.read())
    f.close()
    a(length)
filelen("sample.txt",fun)
filelen("test.txt",fun)
```

```
The length of the file is: 5
The length of the file is: 274
```

Problem Statement 4: Write a program using callback mechanism to

1. Sort the list based on the name and display.
2. Sort the list based on the total of P C M marks in descending order and display.

s = [("890", "ram", (95,78,99)), ("123", "kishan", (90,98,89)), ("567", "arjun", (59,68,100))]

Solution:

```
s = [("890","ram",(95,78,99)),("123","kishan",(90,98,89)),
     ("567","arjun",(59,68,100))]

def get_name(t):
    return t[1]

name = sorted(s, key = get_name)
#name = sorted(s, key = lambda t : t[1])
print("Student list is sorted based on 2nd field, Name:\n", name)
```

```
Student list is sorted based on 2nd field, Name:
 [('567', 'arjun', (59, 68, 100)), ('123', 'kishan', (90, 98, 89)), ('89
0', 'ram', (95, 78, 99))]
```

```
s = [("890","ram",(95,78,99)),("123","kishan",(90,98,89)),
     ("567","arjun",(59,68,100))]

def get_totalmarks(t):
    return t[2][0]+t[2][1]+t[2][2]

marks = sorted(s, key = get_totalmarks)
print("Student list is sorted based on totalmarks:\n", marks)
```

```
Student list is sorted based on totalmarks:
 [('567', 'arjun', (59, 68, 100)), ('890', 'ram', (95, 78, 99)), ('123',
'kishan', (90, 98, 89))]
```

Day: Wednesday
Sections: C1, K1, J2, F2

Problem Statement 1: Write a Python program that takes an integer as input 'n' and counts the number of digits in the binary representation of 'n' using recursion.
Solution:

```
def count_binary_digits(n):
    if n <= 1:
        return 1
    else:
        return 1 + count_binary_digits(n // 2)
n = int(input("Enter a number: "))
result = count_binary_digits(n)
print(result)
```

```
Enter a number: 7
3
```

Problem Statement 2:

Write a recursive function to find the greatest common divisor (GCD) of two positive integers 'a' and 'b', using Euclidian algorithm.

3. Division Method
4. Subtraction Method

Solution: Division method

```
def gcd(a,b):
    if b == 0:
        return a
    else:
        return gcd(b,a%b)
print(gcd(20,12))
```

```
4
```

Solution: Subtraction method

```
#Euclidean algorithm by subtraction
def gcd(a,b):
    if a == b:
        res = a
    elif a>b:
        res = gcd(a-b,b)
    else:
        res = gcd(a,b-a)
    return res
print(gcd(24,87))
```

```
3
```

Problem Statement 3: Write a callback function to display the length (no of characters) in the files "sample.txt" and "test.txt".
Note: Create the files with some content.

```
def fun(l):
    print("The length of the file is:",l)
def filelen(path,a):
    f = open(path,"r")
    length = len(f.read())
    f.close()
    a(length)
filelen("sample.txt",fun)
filelen("test.txt",fun)
```

```
The length of the file is: 5
The length of the file is: 274
```

Problem Statement 4: Write a program using callback mechanism to

1. Sort the list based on the name and display.
2. Sort the list based on the total of P C M marks in descending order and display.

s = [("890", "ram", (95,78,99)), ("123", "kishan", (90,98,89)), ("567", "arjun", (59,68,100))]

Solution:

```
s = [("890","ram",(95,78,99)),("123","kishan",(90,98,89)),
     ("567","arjun",(59,68,100))]

def get_name(t):
    return t[1]

name = sorted(s, key = get_name)
#name = sorted(s, key = lambda t : t[1])
print("Student list is sorted based on 2nd field, Name:\n", name)
```

```
Student list is sorted based on 2nd field, Name:
 [('567', 'arjun', (59, 68, 100)), ('123', 'kishan', (90, 98, 89)), ('89
0', 'ram', (95, 78, 99))]
```

```
s = [("890","ram",(95,78,99)),("123","kishan",(90,98,89)),
     ("567","arjun",(59,68,100))]

def get_totalmarks(t):
    return t[2][0]+t[2][1]+t[2][2]

marks = sorted(s, key = get_totalmarks)
print("Student list is sorted based on totalmarks:\n", marks)
```

```
Student list is sorted based on totalmarks:
 [('567', 'arjun', (59, 68, 100)), ('890', 'ram', (95, 78, 99)), ('123',
'kishan', (90, 98, 89))]
```

Day: Thursday
Sections: B1, N1, O1, D1, M1

Problem Statement 1: Write a recursive function to find the sum of all elements in a list of Integers. Take list of integers as user input.

Solution:

```
def rec_sum(arr):
    if len(arr) == 1:
        return arr[0]
    else:
        return arr[0] + rec_sum(arr[1:])
input_numbers = input("Enter a list of numbers separated by comma:")
number_list = input_numbers.split(",")
numbers = []
for number in number_list:
    numbers.append(int(number))
result = rec_sum(numbers)
print(result)
```

```
Enter a list of numbers separated by comma:10,20,30,40,50
150
```

Problem Statement 2: Write a function that takes a list of numbers and a callback function to find the sum of the numbers based on given conditions:

i. Sum of all even numbers in the list.

ii. Sum of all numbers not divisible by 3 in the list.

Solution:

```
def conditional_sum(numbers, callback):
    total = 0
    for num in numbers:
        if callback(num):
            total += num
    return total
def is_even(x):
    return x % 2 == 0
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = conditional_sum(numbers, is_even)
print(result)  # Output: 30 (since 2 + 4 + 6 + 8 + 10 = 30)
```

```python
def conditional_sum(numbers, callback):
    total = 0
    for num in numbers:
        if callback(num):
            total += num
    return total
def not_divisible_by_3(x):
    return x % 3 != 0
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = conditional_sum(numbers, not_divisible_by_3)
print(result)  # Output: 54 (since 6 + 12 + 15 + 18 + 24 = 75)
```

Problem Statement 3: Write a python function that takes a list of strings as a callback function. It should apply the callback to each string in the list, but only return the results for strings that meet the following condition:

i) Find all the strings starting with Letter 'A'

ii) Find all the strings whose length is greater than 4.

Solution:

```python
def callback(strings, callback):
    results = []
    for string in strings:
        if callback(string):
            results.append(string)
    return results
def starts_with_a(string):
    return string.startswith('A')
def length_four(string):
    return len(string) > 4
strings = ['Antarctica', 'America', 'Boston', 'Canada', 'Africa', 'Pesu', 'Ben','Aero']
result = callback(strings, starts_with_a)
print(result)  #['Antarctica', 'America', 'Africa', 'Aero']
result = callback(strings, length_four)
print(result) #['Antarctica', 'America', 'Boston', 'Canada', 'Africa']
```

Problem Statement 4: Write a Python program that takes an integer 'n' as input and determines whether it is a power of 2. The program should return True if the input number is a power of 2 or False otherwise using recursion

Solution:

Solution:

```
def is_power_of_two(n):
    if n <= 0:
        return False
    elif n == 1:
        return True
    elif n % 2 == 0:
        return is_power_of_two(n // 2)
    else:
        return False

# Input from the user
num = int(input("Enter a number: "))
result = is_power_of_two(num)

if result:
    print("True")
else:
    print("False")
```

Sample Input and Output:

1) Sample Input 1: 16
   Sample Output 1: True
2) Sample Input 2: 18
   Sample Output 2: False

Day: Friday
Sections: E1, L1, J1, F1, A1, I1

Problem Statement 1: Write a Python program that takes an integer as input 'n' and counts the number of digits in the binary representation of 'n' using recursion.
Solution:

```
def count_binary_digits(n):
    if n <= 1:
        return 1
    else:
        return 1 + count_binary_digits(n // 2)
n = int(input("Enter a number: "))
result = count_binary_digits(n)
print(result)
```

```
Enter a number: 7
3
```

Problem Statement 2:

Write a recursive function to find the greatest common divisor (GCD) of two positive integers 'a' and 'b', using Euclidian algorithm.

5. Division Method
6. Subtraction Method

Solution: Division method

```
def gcd(a,b):
    if b == 0:
        return a
    else:
        return gcd(b,a%b)
print(gcd(20,12))
```

```
4
```

Solution: Subtraction method

```
#Euclidean algorithm by subtraction
def gcd(a,b):
    if a == b:
        res = a
    elif a>b:
        res = gcd(a-b,b)
    else:
        res = gcd(a,b-a)
    return res
print(gcd(24,87))
```

```
3
```

Problem Statement 3: Write a callback function to display the length (no of characters) in the files "sample.txt" and "test.txt".
Note: Create the files with some content.

```
def fun(l):
    print("The length of the file is:",l)
def filelen(path,a):
    f = open(path,"r")
    length = len(f.read())
    f.close()
    a(length)
filelen("sample.txt",fun)
filelen("test.txt",fun)
```

```
The length of the file is: 5
The length of the file is: 274
```

Problem Statement 4: Write a program using callback mechanism to

1. Sort the list based on the name and display.
2. Sort the list based on the total of P C M marks in descending order and display.

s = [("890", "ram", (95,78,99)), ("123", "kishan", (90,98,89)), ("567", "arjun", (59,68,100))]

Solution:

```
s = [("890","ram",(95,78,99)),("123","kishan",(90,98,89)),
     ("567","arjun",(59,68,100))]

def get_name(t):
    return t[1]

name = sorted(s, key = get_name)
#name = sorted(s, key = lambda t : t[1])
print("Student list is sorted based on 2nd field, Name:\n", name)
```

```
Student list is sorted based on 2nd field, Name:
 [('567', 'arjun', (59, 68, 100)), ('123', 'kishan', (90, 98, 89)), ('89
0', 'ram', (95, 78, 99))]
```

```
s = [("890","ram",(95,78,99)),("123","kishan",(90,98,89)),
     ("567","arjun",(59,68,100))]

def get_totalmarks(t):
    return t[2][0]+t[2][1]+t[2][2]

marks = sorted(s, key = get_totalmarks)
print("Student list is sorted based on totalmarks:\n", marks)
```

```
Student list is sorted based on totalmarks:
 [('567', 'arjun', (59, 68, 100)), ('890', 'ram', (95, 78, 99)), ('123',
'kishan', (90, 98, 89))]
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 11

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Programs on functional programming constructs
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

20-11-2023 (Monday):

Sections - L,X (08:00 AM to 9:30 AM):

1) **Problem Statement:** Write a Python function that takes a list of numbers and uses the map function along with a lambda expression to get factorial of each #element in the list.

Solution:

```
from math import factorial

def factorial_of_elements(numbers):
    factorials = list(map(lambda x: factorial(x), numbers))
    return factorials

# Example usage:
lst = []
n = int(input())
for i in range(n):
    ele = int(input())
    lst.append(ele)

result = factorial_of_elements(lst)
print(sum(result))
```

2) Problem Statement: Implement a function that calculates the product of all elements in a list using the reduce function and a lambda expression.

Solution:

```
from functools import reduce

def product_of_elements(numbers):
    product = reduce(lambda x, y: x * y, numbers)
    return product

# Example usage:
lst = []
n = int(input())
for i in range(n):
    ele = int(input())
    lst.append(ele)
result = product_of_elements(lst)
print(result)
```

3) Problem Statement: Create a function that filters out even numbers from a given list using the filter function and a lambda expression.

Solution:

```
def filter_out_even(numbers):
    odd_numbers = list(filter(lambda x: x % 2 == 0, numbers))
    return odd_numbers

# Example usage:
lst = []
n = int(input())
for i in range(n):
    ele = int(input())
    lst.append(ele)
result = filter_out_even(lst)
print(sum(result))
```

Bonus Question:

4) Problem Statement: Write a program that takes a list of numbers, squares each number, and then calculates the product of all squared numbers.

Solution:

```
from functools import reduce

# Function to square a number
square = lambda x: x**2
```

```
# Function to calculate the product of a list of numbers
product = lambda x, y: x * y

# Example usage:
lst = []
n = int(input())
for i in range(n):
   ele = int(input())
   lst.append(ele)

# Use map to square each number in the list
squared_numbers = list(map(square, lst))

# Use reduce to calculate the product of squared numbers
result = reduce(product, squared_numbers)

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

20-11-2023 (Monday):
Sections - C, O (11:30 AM to 1:00 PM):

1) **Problem Statement:** Write a Python program that takes a series of words as input, separated by spaces. Form a list using the 'map' function and then filter out words with more than 3 characters using the 'filter' function along with a 'lambda' function. Finally, print the filtered list in lexicographical order.

   Solution:

```
input_words = input()
word_list = list(map(str, input_words.split()))
filtered_list = list(filter(lambda word: len(word) > 3, word_list))
sorted_list = sorted(filtered_list)
print(sorted_list)
```

2)Problem Statement: Write a Python program that takes a series of words as input, separated by spaces. Form a list using the 'map' function and capitalize the first letter of

each word using 'map' and a 'lambda' function. Finally, print the capitalized list in lexicographical order.

Solution:

```
input_words = input()
word_list = list(map(str, input_words.split()))
capitalized_words = list(map(lambda x: ' '.join(word.capitalize() for word in x.split()), word_list))
print(sorted(capitalized_words))
```

3)Problem Statement: Write a Python program that receives a series of numbers as input, separated by spaces. Form a list using the 'map' function and find the product of elements in the list using 'reduce' and a 'lambda' function. Finally, print the product.

Solution:

```
from functools import reduce

input_numbers = input()
numbers_list = list(map(int, input_numbers.split()))
product = reduce(lambda x, y: x * y, numbers_list)
print(product)
```

Bonus Question:

1) **Problem Statement:** Write a Python function that takes a string as input, where sentences are separated by commas. Using the map function, form a list of sentences and calculate the average length of words in sentences that have an odd number of words. Ensure the result is rounded off to two decimal places.

   Solution:

```
# Import the 'reduce' function from the 'functools' module
from functools import reduce

# Get the input string containing sentences separated by commas
sentence_str = input()

# Split the input string into a list of sentences using commas as the delimiter
sentences = sentence_str.split(',')
```

```
# Use map, filter, and lambda functions to calculate the lengths of words in sentences with an odd number of words
word_lengths = list(map(lambda sentence: list(map(len, sentence.split())), filter(lambda x: len(x.split()) % 2 != 0, sentences)))

# Use reduce to sum up the lengths of all words in qualifying sentences
total_length = reduce(lambda x, y: x + y, [length for sentence_lengths in word_lengths for length in sentence_lengths])

# Calculate the average word length by dividing the total length by the total number of words
average = total_length / sum(len(sentence_lengths) for sentence_lengths in word_lengths)

# Print the average with two decimal places using the 'format' method
print(round(average,2))
```

20-11-2023 (Monday):
Sections - B, N (01:45 AM to 3:15 PM);

1) **Problem Statement:** Write a Python program that takes a series of space-separated numbers as input. Form a list using the 'map' function and filter out numbers from the list that are divisible by either 3 or 5 using the 'filter' function and a 'lambda' function. Finally, print the filtered list.
   Solution:

```
input_numbers = input()
numbers_list = list(map(int, input_numbers.split()))
filtered_numbers = list(filter(lambda x: x % 3 != 0 and x % 5 != 0, numbers_list))
print(filtered_numbers)
```

2) **Problem Statement:** Using map and a lambda function, create a list of squares for odd numbers and cubes for even numbers from the range

Solution:

```
input_numbers = input()
numbers_list = list(map(int, input_numbers.split()))
result = list(map(lambda x: x**2 if x % 2 != 0 else x**3, numbers_list))
print(result)
```

3) **Problem Statement:** Write a Python program that takes a series of words as input, separated by spaces. Use the 'reduce' function to concatenate these words into a single string. Print the resulting concatenated string.

Solution:

```
from functools import reduce

input_words = input()
words_list = input_words.split()
result = reduce(lambda x, y: x + '' + y, words_list)
print(result)
```

Bonus Question:

1) **Problem Statement:** Write a Python program that takes a series of space-separated numbers as input. Form a list using the 'map' function and find the product of squares of even numbers in the list using lambda, map, filter and reduce.

Solution:

```
from functools import reduce

# Input: Accept a series of space-separated numbers
input_numbers = input()

# Convert the input string to a list of integers
numbers_list = list(map(int, input_numbers.split()))

# Use filter to get only even numbers
even_numbers = filter(lambda x: x % 2 == 0, numbers_list)

# Use map to square each even number
squared_numbers = map(lambda x: x**2, even_numbers)
```

```
# Use reduce to find the product of squared even numbers
result = reduce(lambda x, y: x * y, squared_numbers)

# Print the result
print(result)
```

21-11-2023 (Tuesday):

Sections - F, R (8:00AM to 10:30AM)

1) **Problem Statement:** Check if a string is a palindrome using reduce and lambda.
   Solution:

```
from functools import reduce

def is_palindrome(s):
    s = ''.join(filter(str.isalnum, s.lower()))  # Remove non-alphanumeric characters
    reversed_str = reduce(lambda acc, char: char + acc, s, '')
    return s == reversed_str

print(is_palindrome(input("Enter a string: ")))
```

2) **Problem Statement:**Calculate cumulative sum of a list using map

```
def cumulative_sum(nums):
    return [sum(nums[:i+1]) for i in range(len(nums))]

print(cumulative_sum(list(map(int, input().split()))))
```

3) **Problem Statement:** Implement a higher-order function that returns a function that computes the nth power of a number using functional programming using reduce and lambda functions.
   Solution:

```
from functools import reduce

def nth_power_function(n):
    return lambda x: reduce(lambda acc, _: acc * x, range(n), 1)
```

```
base_number = float(input())
power = int(input())

power_function = nth_power_function(power)
result = power_function(base_number)
print(result)
```

4) **(BONUS) Problem Statement:** Count the frequency of each word in a given sentence using functional programming using lambda and reduce functions, given an inputted sentence apply functional programming to find each word's frequency in the statement.

Solution:

```
from functools import reduce

sentence = input()

word_frequency = dict(
   reduce(
      lambda acc, word: {**acc, word: acc.get(word, 0) + 1},
      filter(lambda x: x.isalnum(), map(str.lower, sentence.split())), {}
   )
)

print(word_frequency)
```

21-11-2023 (Tuesday):

Sections - A, M (11:30AM to 1:00PM)

1) **Problem Statement:** Calculate the sum of squares of a list of numbers using map and lambda.

Solution:

```
def sum_of_squares(nums):
   return sum(map(lambda x: x**2, nums))

print(sum_of_squares(list(map(int, input().split()))))
```

2) **Problem Statement:** Find the factorial of a given number using reduce and lambda.
Solution:

```
from functools import reduce

def factorial(n):
   return reduce(lambda x, y: x * y, range(1, n+1), 1)

print(factorial(int(input("Enter a number: "))))
```

3) **Problem Statement:**Filter even numbers from a list using filter and lambda

```
def filter_even(nums):
   return list(filter(lambda x: x % 2 == 0, nums))

print(filter_even(list(map(int, input().split()))))
```

4) **(BONUS) Problem Statement:** Implement a custom map function that can do exactly what the normal map does without using map, implement it for 2* each number in an inputted list
Solution:

```
def custom_map(func, items):
   return [func(item) for item in items]

print(custom_map(lambda x: x * 2, list(map(int, input().split()))))
```

21-11-2023 (Tuesday):
Sections - D, P (1:45PM to 3:15PM)

1) **Problem Statement:** Write a Python function that takes a list of numbers and uses the map function along with a lambda expression to square each element in the list.
Solution:

```
def square_elements(numbers):
   squared_numbers = list(map(lambda x: x**2, numbers))
   return squared_numbers

# Example usage:
lst = []
n = int(input())
```

```
for i in range(n):
    ele = int(input())
    lst.append(ele)

result = square_elements(lst)
print(sum(result))
```

2) **Problem Statement:** Implement a function that calculates the sum of all elements in a list using the reduce function and a lambda expression.

Solution:

```
    from functools import reduce

def sum_of_elements(numbers):
    total_sum = reduce(lambda x, y: x + y, numbers)
    return total_sum

# Example usage:
lst = []
n = int(input())
for i in range(n):
    ele = int(input())
    lst.append(ele)
result = sum_of_elements(lst)
print(result)
```

3) **Problem Statement:** Create a function that filters out even numbers from a given list using the filter function and a lambda expression.

Solution:

```
def filter_out_even(numbers):
    odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
    return odd_numbers

# Example usage:
lst = []
n = int(input())
for i in range(n):
    ele = int(input())
    lst.append(ele)
result = filter_out_even(lst)
```

```
print(sum(result))
```

4) **(BONUS) Problem Statement:** Write a Python function that takes a list of strings and uses the filter function along with a lambda expression to return a new list #containing only the strings that have more than 5 characters.

Solution:

```
from operator import mul
from functools import reduce

def filter_long_numbers(numbers):
    long_numbers = list(filter(lambda s: s > 5, numbers))
    return long_numbers

# Example usage:
lst = []
n = int(input())
for i in range(n):
    ele = int(input())
    lst.append(ele)
result = filter_long_numbers(lst)
r = reduce(mul, result)
print(r)
```

22-11-2023 (Wednesday):
Sections - Y (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program that takes a list of word and filters out words with less than four characters.

Solution:

```
words = input().split()
filtered_words = list(filter(lambda x: len(set(x)) > 3, words))
print(filtered_words)
```

2) **Problem Statement:** Write a program that takes a list of integers as input and finds the sum of the squares of odd numbers from the list. Use map, reduce and filter.

Solution:

```
from functools import reduce

numbers = list(map(int, input().split()))
odd_nums = list(filter(lambda x: x % 2 == 1, numbers))
result = reduce(lambda x, y: x + y, map(lambda x: x**2, odd_nums))

print(result)
```

3) **Problem Statement:** Write a program that takes two lists of integers as input and creates a new list by taking the product of corresponding elements and switches the sign of the product using map function. ie. List3[i] = -(List1[i] * List2[i])

Solution:

```
def product(a, b):
   return -(a * b)

n = list(map(int, input().split()))
m = list(map(int, input().split()))
d = list(map(product, n, m))
print(d)
```

Bonus Question:

1) **Problem Statement:** Write a Python function that takes a string as input, where sentences are separated by commas. Using the map function, form a list of sentences and calculate the average length of words in sentences that have an odd number of words. Ensure the result is rounded off to two decimal places.

Solution:

```
# Import the 'reduce' function from the 'functools' module
from functools import reduce

# Get the input string containing sentences separated by commas
sentence_str = input()
```

```
# Split the input string into a list of sentences using commas as the delimiter
sentences = sentence_str.split(',')

# Use map, filter, and lambda functions to calculate the lengths of words in sentences with an odd number of words
word_lengths = list(map(lambda sentence: list(map(len, sentence.split())), filter(lambda x: len(x.split()) % 2 != 0, sentences)))

# Use reduce to sum up the lengths of all words in qualifying sentences
total_length = reduce(lambda x, y: x + y, [length for sentence_lengths in word_lengths for length in sentence_lengths])

# Calculate the average word length by dividing the total length by the total number of words
average = total_length / sum(len(sentence_lengths) for sentence_lengths in word_lengths)

# Print the average with two decimal places using the 'format' method
print(round(average,2))
```

23-11-2023 (Thursday):

Sections - I, U (8:00AM to 9:30 AM)

1) **Problem Statement:** Write a program that takes a list of integers and converts it into a list containing the number of digits for each integer.

   Solution:

```
n = int(input())
l = list(map(int, input().split()))
def get_digits(x):
    return len(str(x))
l = list(map(get_digits, l))
print(l)
```

2) **Problem Statement:** Write a program that takes a list of strings as input and converts to a list with only strings with length < 5 and which are palindromes.

Solution:

```
def check(x):
   return x==x[::-1] and len(x)<5
n=int(input())
l=input().split()
filtered=list(filter(check,l))
print(filtered)
```

3) **Problem Statement:** Write a program to take a list of strings and concatenate the strings and convert it to upper case.

Solution:

```
from functools import reduce
def concatenate_and_uppercase(result, string):
   return (result + string).upper()
n=int(input())
strings_list =input().split()
result = reduce(concatenate_and_uppercase, strings_list, "")
print(result)
```

Bonus Question:

4) **Problem Statement:** Write a program that takes a string as input and create a dictionary with the frequency of the alphabets in the string. Use map, filter and reduce functions.

Solution:

```
from functools import reduce
def update_char_frequency(char_frequency, char):
   char_lower = char.lower()
   char_frequency[char_lower] = char_frequency.get(char_lower, 0) + 1
   return char_frequency

input_string = input().lower()
characters = list(map(lambda x: x, input_string))
```

```
alphabetic_characters = list(filter(lambda x: x.isalpha(), characters))
char_frequency_dict = reduce(update_char_frequency, alphabetic_characters, {})
print(char_frequency_dict)
```

23-11-2023 (Thursday):
Sections - E, Q (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program that takes a list of sentences and converts it to a list containing the number of words in each sentence that starts with 'S'. For example "shining so bright" has 2 words starting with the letter 's'

Solution:

```
n = int(input())
l = input().split('-')
def get_words(x):
    x=x.split()
    cnt=0
    for i in x:
        if i[0].lower()=='s':
            cnt+=1
    return cnt
l = list(map(get_words, l))
print(l)
```

2) **Problem Statement:** Write a program that takes a list of floating-point numbers and converts it to a list with only the numbers which have more than two decimal places

Solution:

```
def check_decimal(number):
    decimal_part = str(number).split('.')[1]
    return len(decimal_part) > 2
n=int(input())
l=list(map(float,input().split()))
filtered=list(filter(check_decimal,l))
print(filtered)
```

3) **Problem Statement:** Write a program to take a list of strings and return the total count of vowels

Solution:

```
from functools import reduce
def count_vowels(count, word):
   vowels = "aeiouAEIOU"
   return count + sum(1 for char in word if char in vowels)
n=int(input())
words_list = input().split()
total_vowels_count = reduce(count_vowels, words_list, 0)
print(total_vowels_count)
```

Bonus Question:

4) **Problem Statement:** Write a program that takes a list of strings as input, identifies the palindromes among them, and creates a dictionary where the palindromes are keys, and their lengths are values.

Solution:

```
from functools import reduce
def is_palindrome(word):
   return word == word[::-1]

def update_palindrome_length(palindrome_dict, word):
   palindrome_length = len(word)
   palindrome_dict[word] = palindrome_length
   return palindrome_dict
n=int(input())
words_list = input().split()
palindromes = list(filter(is_palindrome, words_list))
palindrome_length_dict = reduce(update_palindrome_length, palindromes, {})
print(palindrome_length_dict)
```

23-11-2023 (Thursday):
Sections - G, S (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program that takes two lists as input, each containing integer elements. Convert it to a list that contains True if the product of corresponding elements from both lists is divisible by 5, and False otherwise.

Solution:

```
def list_check(x, y):
   return True if (x * y) % 5==0 else False

list1 = list(map(int,input().split()))
list2 = list(map(int,input().split()))

result = list(map(list_check, list1, list2))

print(result)
```

2) **Problem Statement:** Write a program to get the sum of cubes of odd numbers in a list of integers. Do not use functions, use lambda instead.

Solution:

```
from functools import reduce
l=list(map(int,input().split()))
l=list(filter(lambda x: x%2!=0,l))
l=list(map(lambda x:x**3,l))
result=reduce(lambda x,y:x+y,l,0)
print(result)
```

3) **Problem Statement:** Write a python program that converts a list of temperatures in Celsius to Fahrenheit using the map function and a lambda expression.
(Hint: the conversion formula is °F = °C × (9/5) + 32)

Solution:

```
celsius_to_fahrenheit = lambda c: (c * 9/5) + 32

n = int(input())
celsius_temperatures =[]
for i in range(n):
   ele = int(input())
   celsius_temperatures.append(ele)
```

```
fahrenheit_temperatures = list(map(celsius_to_fahrenheit, celsius_temperatures))
print("Temperatures in Celsius:", celsius_temperatures)
print("Temperatures in Fahrenheit:", fahrenheit_temperatures)
```

Bonus Question:

4) **Problem Statement:** Implement a function that filters out all prime numbers from a given list of integers using the filter function and a lambda expression.

Solution:

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def filter_prime_numbers(numbers):
    return list(filter(lambda x: is_prime(x), numbers))

n = int(input())
input_numbers =[]
for i in range(n):
    ele = int(input())
    input_numbers.append(ele)
filtered_primes = filter_prime_numbers(input_numbers)
print("Filtered prime numbers:", filtered_primes)
```

24-11-2023 (Friday):
Sections - J, V (8:00AM to 9:30 AM)

1) **Problem Statement:** Write a program to take a list of strings and return the total count of consonants.

Solution:

```
from functools import reduce

def count_vowels(count, word):
```

```
    vowels = "aeiouAEIOU"
    return count + sum(1 for char in word if char not in vowels)

words_list = input().split()
total_vowels_count = reduce(count_vowels, words_list, 0)
print(total_vowels_count)
```

2) **Problem Statement:** Write a python program that filters a list of strings and returns only the palindromes using the filter function and a lambda expression.

Solution:

```
is_palindrome = lambda s: s == s[::-1]

n = int(input())
input_strings =[]
for i in range(n):
    ele = input()
    input_strings.append(ele)

palindromes = list(filter(is_palindrome, input_strings))
print("List of palindromes:", palindromes)
```

3) **Problem Statement:** Implement a program that takes two lists of integers, multiplies corresponding elements together using lambda, and creates a new list with the multiplied values.

Solution:

```
n = int(input())
list1 = [ int(input()) for _ in range(n)]
list2 = [int(input()) for _ in range(n)]

multiply_lists = lambda x, y: [a * b for a, b in zip(x, y)]
result_list = multiply_lists(list1, list2)

print("List 1:", list1)
print("List 2:", list2)
print("List after Multiplication:", result_list)
```

Bonus Question:

4) **Problem Statement:** Create a function that takes a list of sentences and returns a list containing the lengths of each word in each sentence using map() and lambda

Solution:

```
def word_lengths(sentences):
  # Split each sentence into words and map to their lengths
    word_lengths_list = list(map(lambda sentence: list(map(len, sentence.split())), sentences))
  return word_lengths_list

n = int(input())
sentences = [input() for _ in range(n)]
result = word_lengths(sentences)
print(result)
```

24-11-2023 (Friday):
Sections - K, W (11:30AM to 1:00PM)

1) **Problem Statement:** Write a program that takes a sentence as input, capitalizes the first letter of every word and concatenates them.

Solution:

```
from functools import reduce

def concatenate_cap(result, string):
  return (result + string.capitalize())

words = input().split()
result = reduce(concatenate_cap, words, "")
print(result)
```

2) **Problem Statement:** Write a program that takes two lists of integers as input and creates a new list by taking the product of corresponding elements using map function. ie. List3[i] = List1[i] * List2[i]

Solution:

```
def product(a, b):
    return a * b

n = list(map(int, input().split()))
m = list(map(int, input().split()))
p = list(map(product, n, m))
print(p)
```

3) **Problem Statement:** Write a program that takes a list of floating-point numbers and creates a new list such that each number is rounded off to the nearest integer and all two digit numbers are filtered out.

Solution:

```
f = list(map(float, input().split()))
rounded = list(map(round, f))
two_digits = list(filter(lambda n: not(10 <= n <= 99), rounded))
print(two_digits)
```

Bonus Question:

4) **Problem Statement:** Write a program that takes a list of words as input and prints the longest word using reduce function. If two words have the same length, take the lexicographically bigger word.
Order of precedence - length(word), lower case letters, upper case letters.

Solution:

```
from functools import reduce

def longest(a, b):
    if len(a) > len(b):
        return a
    elif len(a) < len(b):
        return b
    else:
        # Same length, compare characters
        # Since ASCII value of lowercase characters is more than upper
        # We will swap cases to compare them
```

```
    if a.swapcase() > b.swapcase():
        return a
    else:
        return b

words = list(input().split())
l = reduce(longest, words, words[0])
print(l)
```

24-11-2023 (Friday):
Sections - H, T (1:45PM to 3:15PM)

1) **Problem Statement:** Write a program that takes a sentence as input, converts all the words to lowercase and concatenates them.

Solution:

```
from functools import reduce

def concatenate_low(result, string):
    return (result + string.lower())

words = input().split()
result = reduce(concatenate_low, words, "")
print(result)
```

2) **Problem Statement:** Write a program that takes two lists of integers as input and creates a new list by taking the difference of corresponding elements using map function. ie. List3[i] = List1[i] - List2[i]

Solution:

```
def difference(a, b):
    return a - b

n = list(map(int, input().split()))
m = list(map(int, input().split()))
d = list(map(product, n, m))
print(d)
```

3) **Problem Statement:** Write a program that takes a list of positive and negative integers and creates a new list such that each number is converted to its absolute value and all numbers containing more than 2 digits are filtered out.

Solution:

```
n = list(map(int, input().split()))
absolute = list(map(abs, n))
two_and_less = list(filter(lambda n: n <= 99, absolute))
print(two_and_less)
```

Bonus Question:

4) **Problem Statement:** Write a program that takes a list of words as input and prints the smallest word using reduce function. If two words have the same length, take the lexicographically smaller word.
Order of precedence - length(word), lower case letters, upper case letters

Solution:

```
from functools import reduce

def smallest(a, b):
    if len(a) < len(b):
        return a
    elif len(a) > len(b):
        return b
    else:
        # Same length, compare characters
        # Since ASCII value of lowercase characters is more than upper
        # We will swap cases to compare them to satisfy required condition
        if a.swapcase() < b.swapcase():
            return a
        else:
            return b

words = list(input().split())
s = reduce(smallest, words, words[0])
print(s)
```

****

Python for Computational Problem-Solving

LABORATORY MANUAL

Week 11

Topic: Functional Programming Constructs

Semester: I
Course Code: UE23CS151A
Course Anchor: Prof. Sindhu R Pai
Lab Anchor: Dr. Divyashree N
Anchor and Lab Anchor – EC Campus: Prof. Kundhavai K R
Session: September 2023 – December 2023

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab session unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To Learn and Solve	• Programs on Functional Programming constructs
	• Program solutions for mandatory questions should be submitted for evaluation. • Additional programs are only for practice, and not for grading

(Submissions required. Students should submit their codes through hacker rank platform)

Day: Monday
Sections: O2, C2, K2, B2, N2

Problem Statement 1:
You've started a garden with plants that grow at different rates. Write a Python function calculate_growth using the map function to estimate the height of each plant after a certain number of days.
Height = initial height + growth_rate * no_of_days

Input: Two lists - one representing the initial heights of plants and the other representing the growth rates per day.
Output: A list of floating-point numbers representing the estimated heights after a specified number of days.

Solution:

```
def calculate_growth(initial_heights, growth_rates, days):
    return list(map(lambda x, y: x + y * days, initial_heights, growth_rates))

# Test
initial_heights = [10.0, 15.0, 8.0, 12.0]
growth_rates = [0.5, 0.8, 0.3, 0.6]
days = 5
estimated_heights = calculate_growth(initial_heights, growth_rates, days)
print(estimated_heights)
```

Output:

```
[12.5, 19.0, 9.5, 15.0]
```

Problem Statement 2:
You're organising a coding competition and want to filter out participants who solved a specific type of problem.
Write a Python program filter_problem_solvers using the filter function to select only those who solved a given problem.

Input: A list of dictionaries representing participants, each containing 'name', 'id', and 'solved_problems'.
Output: A list of dictionaries representing participants who solved the specified problem.

Solution:

```
def filter_problem_solvers(participants, target_problem):
    return list(filter(lambda x: target_problem in x['solved_problems'], participants))

# Test
participants = [{'name': 'Alice', 'id': 1, 'solved_problems': ['A', 'B', 'C']},
                {'name': 'Bob', 'id': 2, 'solved_problems': ['B', 'D', 'E']},
                {'name': 'Charlie', 'id': 3, 'solved_problems': ['A', 'C', 'D']}]
target_problem = 'C'
selected_participants = filter_problem_solvers(participants, target_problem)
print(selected_participants)
```

Output:

```
[{'name': 'Alice', 'id': 1, 'solved_problems': ['A', 'B', 'C']}, {'name': 'Charlie', 'id': 3, 'solved_problems': ['A', 'C', 'D']}]
```

Problem Statement 3:
You're a chef preparing a multi-course meal and want to calculate the total cooking time. Write a Python program calculate_cooking_time using the reduce function to find the overall cooking time.

Input: A list of integers representing the cooking times for each course in minutes.
Output: An integer representing the total cooking time.

Solution:

```
from functools import reduce

# Sample Code
def calculate_cooking_time(cooking_times):
    return reduce(lambda x, y: x + y, cooking_times)

# Test
cooking_times = [30, 45, 20, 60, 15]
total_cooking_time = calculate_cooking_time(cooking_times)
print(total_cooking_time)
```

Output:

```
170
```

Problem Statement 4:
You're designing a simple encryption system for messages. Write a Python function encrypt_messages using the map function to encrypt a list of messages using a basic Caesar cipher.

Input: A list of strings representing messages and an integer representing the shift value for the Caesar cipher.
Output: A list of strings representing the encrypted messages.

Solution:

```python
def encrypt_messages(messages, shift):
    return list(map(lambda x: ''.join([chr((ord(char) - ord('A') + shift) % 26 + ord('A')) for char in x.upper()]), messages))

# Test
messages = ['Hello', 'Python', 'Computer Science']
shift = 3
encrypted_messages = encrypt_messages(messages, shift)
print(encrypted_messages)
```

Output:

```
['KHOOR', 'SBWKRQ', 'FRPSXWHUWVFLHQFH']
```

Problem Statement 5
Write a Python program that takes user input for two lists of integers, and then prints the squares of the common elements between the two lists. Use functional programming constructs:

Solution:

```python
def common_squares(list1, list2):
    common_elements = filter(lambda x: x in list2, list1)
    square_common_elements = map(lambda x: x**2, common_elements)
    return list(square_common_elements)

# Take input for the first list
list_a = list(map(int, input("Enter elements of the first list separated by space: ").split()))

# Take input for the second list
list_b = list(map(int, input("Enter elements of the second list separated by space: ").split()))

result = common_squares(list_a, list_b)
print(result)
```

Output:

```
Enter elements of the first list separated by space: 2 4 6 8
Enter elements of the second list separated by space: 1 2 3 4
[4, 16]
```

Day: Tuesday
Sections: D2, I2, E2, L2

Problem Statement 1:
You're building a file management system and need to extract file extensions from a list of file names. Write a Python program extract_file_extensions using the map function to obtain a list of file extensions.

Input: A list of strings representing file names.
Output: A list of strings representing file extensions.

Solution:

```
def extract_file_extensions(file_names):
    return list(map(lambda x: x.split('.')[-1], file_names))

# Test
file_names = ['document.txt', 'image.png', 'script.js', 'data.csv']
file_extensions = extract_file_extensions(file_names)
print(file_extensions)
```

Output:

```
['txt', 'png', 'js', 'csv']
```

Problem Statement 2:
You're developing a racing game, and you want to calculate lap times for each player. Write a Python program calculate_lap_times using the map function to determine lap times based on start and finish times.

Input: Two lists - one representing player names and the other representing tuples of start and finish times.
Output: A dictionary where keys are player names, and values are their corresponding lap times.

Solution:

```
def calculate_lap_times(player_names, start_finish_times):
    return dict(map(lambda player, times: (player, times[1] - times[0]), player_names, start_finish_times))

# Test
player_names = ['Alice', 'Bob', 'Charlie']
start_finish_times = [(10, 25), (15, 30), (12, 28)]
lap_times = calculate_lap_times(player_names, start_finish_times)
print(lap_times)
```

Output:

```
{'Alice': 15, 'Bob': 15, 'Charlie': 16}
```

Problem Statement 3:
Write a Python function double_numbers that takes a list of numbers and returns a new list where each number is doubled using map.

Solution:

```
def double_numbers(numbers):
    return list(map(lambda x: x * 2, numbers))
numbers=[1,2,3,4]
print(double_numbers(numbers))
```

Output:

```
[2, 4, 6, 8]
```

Problem Statement 4:

Create a program that reads a list of words from the user, and filters out the words that have less than 4 characters.

Solution:

```
words = input("Enter a list of words separated by space: ").split()


# Filter words with less than 4 characters

filtered_words = list(filter(lambda word: len(word) >= 4, words))
```

```
# Display the filtered list of words

print("Words with 4 or more characters:", filtered_words)Output:
```

Output:

```
Enter a list of words separated by space: hello bye goodday
Words with 4 or more characters: ['hello', 'goodday']
>
```

Problem Statement 5:
Write a Python function that takes the dimensions (number of rows and columns) of a matrix as input, then takes the matrix elements as input, and returns the transposed matrix. Use functional programming constructs.

Solution:

```
def transpose_matrix(matrix):
    # Transpose the matrix using zip and list comprehension
    transposed = [list(row) for row in zip(*matrix)]
    return transposed

# Take input for the matrix dimensions
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Take input for the matrix elements
matrix = []
print("Enter the matrix elements:")
for _ in range(rows):
    row = list(map(int, input().split()))
    matrix.append(row)

# Call the transpose_matrix function
result = transpose_matrix(matrix)

# Print the transposed matrix
print("Transposed Matrix:")
for row in result:
    print(row)
```

Output:

```
Enter the number of rows: 3
Enter the number of columns: 4
Enter the matrix elements:
2 4 6 8
1 2 3 4
4 5 6 7
Transposed Matrix:
[2, 1, 4]
[4, 2, 5]
[6, 3, 6]
[8, 4, 7]
```

Day: Wednesday
Sections: C1, K1, J2, F2

Problem Statement 1:
Use a lambda function to square each element of a given list using the map function.

```
input_numbers = input("Enter a list of numbers separated by space: ").split()
numbers = list(map(float, input_numbers))  # Convert input strings to floats

# Use map and a lambda function to square each element
squared_numbers = list(map(lambda x: x ** 2, numbers))

# Display the squared elements
print("Squared elements:", squared_numbers)
```

Output:

```
Enter a list of numbers separated by space: 10 20 30 50
Squared elements: [100.0, 400.0, 900.0, 2500.0]
>
```

Problem Statement 2:
Create a Python function that filters a list of integers and returns only the even numbers. Implement this using the filter function along with a lambda expression.
Solution:

```
size = int(input("Enter the size of the array: "))
numbers = []
for i in range(size):
    x=float(input(f"enter {i+1} number:"))
    numbers.append(x)



def filter_even_numbers(numbers):
    # Using filter with a lambda function to filter even numbers
```

```
    even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
    return even_numbers



# Call the function and print the result
result = filter_even_numbers(numbers)
print("Even numbers:", result)
```

Output:

```
Enter the size of the array: 5
enter 1 number:10
enter 2 number:11
enter 3 number:12
enter 4 number:13
enter 5 number:14
Even numbers: [10.0, 12.0, 14.0]
>
```

Problem Statement 3:
Implement a Python program to find the maximum element in a list using the reduce function.

Solution:

```
from functools import reduce


size = int(float(input("Enter the size of the array: ")))
numbers = []
for i in range(size):
    x = float(input(f"Enter {i+1} number: "))
    numbers.append(x)

# Function to find the maximum of two numbers
def find_max(a, b):
    return a if a > b else b

# Function to find the maximum element in a list using reduce
```

```
def find_maximum_in_list(lst):
    return reduce(find_max, lst)

# Call the function and print the result
maximum_element = find_maximum_in_list(numbers)
print("Maximum element in the list:", maximum_element)
```

Output:

```
Enter the size of the array: 4
Enter 1 number: 20
3Enter 2 number: 30
Enter 3 number: 40
Enter 4 number: 50
Maximum element in the list: 50.0
```

Problem Statement 4:
Given a list of strings, write a Python program to filter out strings with a length less than 5 and convert the remaining strings to uppercase using map and filter together with lambda functions.

Solution:

```
string_list = ['apple', 'banana', 'grapes', 'orange', 'kiwi', 'pear']

# Filter strings with length less than 5 and convert remaining strings to uppercase
filtered_and_uppercased = list(map(lambda x: x.upper(), filter(lambda x: len(x) >= 5, string_list)))

# Print the filtered and uppercase strings
print("Filtered strings :", filtered_and_uppercased)
```

Output:

```
Filtered strings with length >= 5 and converted to uppercase: ['APPLE', 'BANANA', 'GRAPES',
    'ORANGE']
```

Problem Statement 5:

Write a Python function that takes an integer as input and returns the product of its prime factors. Use functional programming constructs.

Solution:

```
from functools import reduce

def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def prime_factors_product(n):
    prime_factors = filter(lambda x: n % x == 0 and is_prime(x), range(2, n + 1))
    product = reduce(lambda x, y: x * y, prime_factors, 1)
    return product

n=int(input())
print(prime_factors_product(n))
```

Sample input =12

output=6

Day: Thursday
Sections: B1, N1, O1, D1, M1

Problem Statement 1:
Write a Python program that takes a list of integers as input and returns a new list with each element multiplied by 3 using the map function.

Solution:

```
def multiply_by_3(num):
    return num * 3

# Take a list of integers as input from the user
input_numbers = input("Enter a list of integers separated by space: ").split()
numbers = list(map(int, input_numbers))

result = list(map(multiply_by_3, numbers))

# Display the new list with each element multiplied by 3
print("List with each element multiplied by 3:", result)
```

Output:

```
Enter a list of integers separated by space: 1 2 3 4 5
List with each element multiplied by 3: [3, 6, 9, 12, 15]
>
```

Problem Statement 2:
Create a Python function that filters a list of numbers to return only the odd numbers using the filter function.

Solution:

```
def filter_odd_numbers(numbers):
    odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
    return odd_numbers
# Take user input for a list of numbers
```

```
input_numbers = input("Enter a list of numbers separated by space: ").split()
numbers = list(map(int, input_numbers))  # Convert input strings to integers

result = filter_odd_numbers(numbers)

# Display the list with only the odd numbers
print("Odd numbers:", result)
```

Output:

```
Enter a list of numbers separated by space: 1 2 3 4 5
Odd numbers: [1, 3, 5]
>
```

Problem Statement 3:
You are given a list of numeric strings. Convert these strings to their corresponding integer values, filter out numbers greater than 100, and then calculate the product of the remaining numbers using the map(), lambda, and reduce() functions.
numeric_strings = ['50', '120', '90', '80', '105', '95']
Solution:

```
from functools import reduce

numeric_strings = ['50', '120', '90', '80', '105', '95']

# Convert strings to integers
integers = list(map(int, numeric_strings))
# Filter numbers greater than 100
filtered_numbers = list(filter(lambda x: x <= 100, integers))
# Calculate the product of the filtered numbers
product = reduce(lambda x, y: x * y, filtered_numbers)

print("Product of numbers less than or equal to 100:", product)
```

Output:

```
Product of numbers less than or equal to 100: 34200000
```

Problem Statement 4:
You are planning a garden and want to calculate the area of each garden bed. Write a Python function calculate_bed_area that takes a list of side lengths and uses the map function to find the area of each bed.

Input: A list of positive integers representing the side lengths of garden beds.
Output: A list of integers representing the area of each garden bed.

Solution:

```
def calculate_bed_area(side_lengths):
    return list(map(lambda x: x**2, side_lengths))

# Test
side_lengths = [3, 5, 2, 4, 6]
area_results = calculate_bed_area(side_lengths)
print(area_results)
```

Output:

```
[9, 25, 4, 16, 36]
```

Problem Statement 5:
Write a Python function that takes an integer n as input and returns a list containing the squares of even numbers in the first n terms of the Fibonacci sequence. Use functional programming constructs.
Solution:

```
def fibonacci_squares(n):
    fib_sequence = [0, 1]
    while len(fib_sequence) < n:
        fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])

    even_numbers = filter(lambda x: x % 2 == 0, fib_sequence)
    squares_of_even = map(lambda x: x**2, even_numbers)

    return list(squares_of_even)

n=int(input())
result = fibonacci_squares(n)
print(result)
```

Output:

```
5
[0, 4]
```

Day: Friday
Sections: E1, L1, J1, F1, A1, I1

Problem Statement 1:
Write a Python program using a map function that takes a list of temperatures in Celsius from the user and converts them into Fahrenheit. Use a lambda function to perform the conversion.

```
size = int(input("Size of the Array: "))

celsius_temperatures = []

for i in range(size):
    temp = float(input(f"Enter Temperature {i + 1} in Celsius: "))   float
    celsius_temperatures.append(temp)

# Using map and lambda function to convert Celsius to Fahrenheit
fahrenheit_temperatures = list(map(lambda c: (c * 9/5) + 32,
celsius_temperatures))

# Displaying the converted temperatures
print("Celsius Temperatures:", celsius_temperatures)
print("Fahrenheit Temperatures:", fahrenheit_temperatures)
```

Solution:

```
Size of the Array: 5
Enter Temperature 1 in Celsius: 30
Enter Temperature 2 in Celsius: 20
Enter Temperature 3 in Celsius: 24
Enter Temperature 4 in Celsius: 33
Enter Temperature 5 in Celsius: 55
Celsius Temperatures: [30.0, 20.0, 24.0, 33.0, 55.0]
Fahrenheit Temperatures: [86.0, 68.0, 75.2, 91.4, 131.0]
>
```

Problem Statement 2:
You're designing a simple encryption system for messages. Write a Python function encrypt_messages using the map function to encrypt a list of messages using a basic Caesar cipher.

Input: A list of strings representing messages and an integer representing the shift value for the Caesar cipher.
Output: A list of strings representing the encrypted messages.

Solution:

```
def encrypt_messages(messages, shift):
    return list(map(lambda x: ''.join([chr((ord(char) - ord('A') + shift) % 26 + ord('A')) for char in x.upper()]), messages))

# Test
messages = ['Hello', 'Python', 'Computer Science']
shift = 3
encrypted_messages = encrypt_messages(messages, shift)
print(encrypted_messages)
```

Output:

```
['KHOOR', 'SBWKRQ', 'FRPSXWHUWVFLHQFH']
```

Problem Statement 3:
You're developing a word-guessing game. Write a Python program guess_the_word using the filter function to identify words containing a specific letter.

Input: A list of strings representing words and a character representing the target letter.
Output: A list of strings representing words containing the target letter.
Solution:

```
def guess_the_word(words, target_letter):
    return list(filter(lambda x: target_letter in x, words))

# Test
words = ['python', 'java', 'programming', 'code', 'challenge']
target_letter = 'a'
words_with_letter = guess_the_word(words, target_letter)
print(words_with_letter)
```

Output:

```
['java', 'programming', 'challenge']
```

Problem Statement 4:
Develop a Python program that calculates the product of all elements in a given list using the reduce function from the functools module. Ensure to use an appropriate lambda function.

Solution:

```
from functools import reduce

input_numbers = input("Enter a list of numbers separated by space: ").split()
numbers = list(map(float, input_numbers))
# Calculate the product of all elements in the list using reduce and lambda function
product = reduce(lambda x, y: x * y, numbers)
# Display the product of all elements in the list
print("Product of all elements in the list:", product)
```

Output:

```
Enter a list of numbers separated by space: 1 3 4 6 7
Product of all elements in the list: 504.0
>
```

Problem Statement 5:

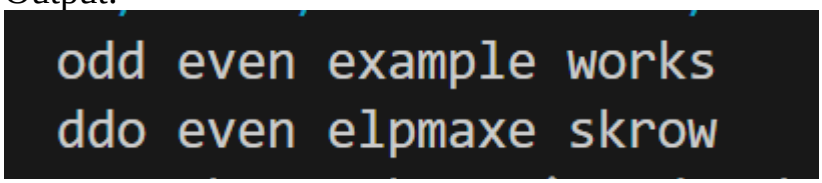
Write a Python function that takes a string as input and returns a new string where the odd-length words are reversed. Words are separated by spaces. Use functional programming constructs.

Solution:

```
def reverse_odd_length_words(sentence):
    words = sentence.split()
    reversed_odd_length_words = map(lambda word: word[::-1] if len(word) % 2 != 0 else word, words)
    return ' '.join(reversed_odd_length_words)

input_sentence = input()
result = reverse_odd_length_words(input_sentence)
print(result)
```

Output:



****
***