# Rajalakshmi Engineering College

Name: gladwin antony.A
Email: 241501057@rajalakshmi.edu.in
Roll no: 241501057
Phone: 8015799480
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 5_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 19

## Section 1 : MCQ

1. Set s1 = {1, 2, 4, 3} and s2 = {1, 5, 4, 6}, find s1 & s2, s1 - s2, s1 | s2 and s1 ^ s2.

***Answer***

s1&s2 = {1, 4}s1-s2 = {2, 3}s1^s2 = {2, 3, 5, 6}s1|s2 = {1, 2, 3, 4, 5, 6}

***Status :*** Correct ***Marks : 1/1***

2. Which of the following isn't true about dictionary keys?

***Answer***

Keys must be integers

***Status :*** Correct ***Marks : 1/1***

3. Which of the following is a Python tuple?

***Answer***

(1, 2, 3)

***Status :*** Correct ***Marks : 1/1***

4. What will be the output for the following code?

```
a=(1,2,3)
b=('A','B','C')
c=zip(a,b)

print(c)
print(tuple(c))
```

***Answer***

((1, 'A'), (2, 'B'), (3, 'C'))

***Status :*** Correct ***Marks : 1/1***

5. Which of the following statements is used to create an empty tuple?

***Answer***

( )

***Status :*** Correct ***Marks : 1/1***

6. What is the result of print(type({}) is set)?

***Answer***

False

***Status :*** Correct ***Marks : 1/1***

7. What will be the output of the following program?

```
set1 = {1, 2, 3}
set2 = set1.copy()
set2.add(4)
print(set1)
```

***Answer***

{1, 2, 3}

***Status :*** Correct ***Marks : 1/1***

8. What is the output of the following code?

```
a=(1,2,(4,5))
b=(1,2,(3,4))
print(a<b)
```

***Answer***

False

***Status :*** Correct ***Marks : 1/1***

9. Suppose t = (1, 2, 4, 3), which of the following is incorrect?

***Answer***

t[3] = 45

***Status :*** Correct ***Marks : 1/1***

10. What is the output of the below Python code?

```
list1 = [1, 2, 3]
list2 = [5, 6, 7]
list3 = [10, 11, 12]
set1 = set(list2)
set2 = set(list1)
set1.update(set2)
set1.update(list3)
print(set1)
```

***Answer***

{1, 2, 3, 5, 6, 7, 10, 11, 12}

***Status :*** Correct ***Marks : 1/1***

11. What will be the output of the following code?

```
a=(1,2,3,4)
print(sum(a,3))
```

***Answer***

13

***Status :*** Correct ***Marks : 1/1***

12. What is the output of the following code?

```
a={1:"A",2:"B",3:"C"}
b=a.copy()
b[2]="D"
print(a)
```

***Answer***

{1: ‘A’, 2: ‘B’, 3: ‘C’}

***Status :*** Correct ***Marks : 1/1***

13. What is the output of the following?

```
set1 = {10, 20, 30, 40, 50}
set2 = {60, 70, 10, 30, 40, 80, 20, 50}
print(set1.issubset(set2))
print(set2.issuperset(set1))
```

***Answer***

FalseFalse

***Status :*** Wrong ***Marks : 0/1***

14. Which of the statements about dictionary values is false?

***Answer***

Values of a dictionary must be unique

***Status :*** Correct ***Marks : 1/1***

15. If 'a' is a dictionary with some key-value pairs, what does a.popitem() do?

***Answer***

Removes an arbitrary element

***Status :*** Correct ***Marks : 1/1***

16. What will be the output?

```
a={'B':5,'A':9,'C':7}
print(sorted(a))
```

***Answer***

['A', 'B', 'C'].

***Status :*** Correct ***Marks : 1/1***

17. Predict the output of the following Python program

```
init_tuple_a = 1, 2, 8
init_tuple_b = (1, 2, 7)
set1=set(init_tuple_b)
set2=set(init_tuple_a)
print (set1 | set2)
print (init_tuple_a | init_tuple_b)
```

***Answer***

{1, 2, 7, 8}TypeError: unsupported operand type

***Status :*** Correct ***Marks : 1/1***

18. What will be the output for the following code?

```
t1 = (1, 2, 4, 3)
t2 = (1, 2, 3, 4)
print(t1 < t2)
```

***Answer***

False

***Status :*** Correct ***Marks : 1/1***

19. Fill in the code in order to get the following output.

Output:

Tuple: (1, 3, 4)

Max value: 4

```
t=(1,)
_____
print("Tuple:" ,t)
print("Max value:",_____)
```

***Answer***

1) t=t+(3,4)2) max(t)

***Status :*** Correct ***Marks : 1/1***

20. What is the output of the following code?

```
a={"a":1,"b":2,"c":3}
b=dict(zip(a.values(),a.keys()))
print(b)
```

***Answer***

{1: ‘a’, 2: ‘b’, 3: ‘c’}

***Status :*** Correct ***Marks : 1/1***

# Rajalakshmi Engineering College

Name: gladwin antony.A
Email: 241501057@rajalakshmi.edu.in
Roll no: 241501057
Phone: 8015799480
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 5_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

## Section 1 : Coding

1. Problem Statement

Liam is analyzing a list of product IDs from a recent sales report. He needs to determine how frequently each product ID appears and calculate the following metrics:

Frequency of each product ID: A dictionary where the key is the product ID and the value is the number of times it appears.Total number of unique product IDs.Average frequency of product IDs: The average count of all product IDs.

Write a program to read the product IDs, compute these metrics, and output the results.

Example

Input:

6 //number of product ID

101

102

101

103

101

102 //product IDs

Output:

{101: 3, 102: 2, 103: 1}

Total Unique IDs: 3

Average Frequency: 2.00

Explanation:

Input 6 indicates that you will enter 6 product IDs.

A dictionary is created to track the frequency of each product ID.

Input 101: Added with a frequency of 1.

Input 102: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 2.

Input 103: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 3.

Input 102: Frequency of 102 increased to 2.

The dictionary now contains 3 unique IDs: 101, 102, and 103.

Total Unique is 3.

The average frequency is 2.00.

***Input Format***

The first line of input consists of an integer n, representing the number of product IDs.

The next n lines each contain a single integer, each representing a product ID.

***Output Format***

The first line of output displays the frequency dictionary, which maps each product ID to its count.

The second line displays the total number of unique product IDs, preceded by "Total Unique IDs: ".

The third line displays the average frequency of the product IDs. This is calculated by dividing the total number of occurrences of all product IDs by the total number of unique product IDs, rounded to two decimal places. It is preceded by "Average Frequency: ".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 6
101
102
101
103
101
102

Output: {101: 3, 102: 2, 103: 1}
Total Unique IDs: 3
Average Frequency: 2.00

***Answer***

```
def analyze_product_ids(product_ids):
    """
    Analyzes a list of product IDs to determine their frequency,
    the total number of unique IDs, and the average frequency.

    Args:
        product_ids: A list of integers representing product IDs.
```

```
    Returns:
        A tuple containing:
        - A dictionary mapping product IDs to their frequencies.
        - The total number of unique product IDs.
        - The average frequency of the product IDs (rounded to two decimal places).
    """
    frequency_dict = {}
    for product_id in product_ids:
        if product_id in frequency_dict:
            frequency_dict[product_id] += 1
        else:
            frequency_dict[product_id] = 1

    total_unique_ids = len(frequency_dict)
    total_occurrences = sum(frequency_dict.values())
    average_frequency = total_occurrences / total_unique_ids if total_unique_ids >
0 else 0.0
    return frequency_dict, total_unique_ids, round(average_frequency, 2)

if __name__ == "__main__":
    n = int(input())
    product_ids = []
    for _ in range(n):
        product_id = int(input())
        product_ids.append(product_id)

    frequency_dict, total_unique_ids, average_frequency =
analyze_product_ids(product_ids)

    print(frequency_dict)
    print(f"Total Unique IDs: {total_unique_ids}")
    print(f"Average Frequency: {average_frequency:.2f}")
```

***Status :*** Correct **Marks : 10/10**

2. Problem Statement

James is managing a list of inventory items in a warehouse. Each item is recorded as a tuple, where the first element is the item ID and the second element is a list of quantities available for that item. James needs to filter

out all quantities that are above a certain threshold to find items that have a stock level above this limit.

Help James by writing a program to process these tuples, filter the quantities from all the available items, and display the results.

Note:

Use the filter() function to filter out the quantities greater than the specified threshold for each item's stock list.

***Input Format***

The first line of input consists of an integer N, representing the number of tuples.

The next N lines each contain a tuple in the format (ID, [quantity1, quantity2, ...]), where ID is an integer and the list contains integers.

The final line consists of an integer threshold, representing the quantity threshold.

***Output Format***

The output should be a single line displaying the filtered quantities, space-separated. Each quantity is strictly greater than the given threshold.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 2
(1, [1, 2])
(2, [3, 4])
2
Output: 3 4

***Answer***

```
def filter_inventory_quantities(inventory_data, threshold):
    """
    Filters quantities from inventory data that are strictly greater than a given threshold.
```

```
    Args:
        inventory_data: A list of strings, where each string represents a tuple
                        in the format '(ID, [quantity1, quantity2, ...])'.
        threshold: An integer representing the quantity threshold.

    Returns:
        A string containing the filtered quantities, space-separated.
    """
    filtered_quantities = []
    for item_str in inventory_data:
        item_str = item_str.strip('()')
        parts = item_str.split(', ', 1)
        if len(parts) == 2:
            try:
                item_id = int(parts[0])
                quantities_str = parts[1].strip('[]')
                if quantities_str:
                    quantities = [int(q.strip()) for q in quantities_str.split(',')]
                    filtered = filter(lambda q: q > threshold, quantities)
                    filtered_quantities.extend(list(filtered))
            except ValueError:
                # Handle cases with invalid integer formats
                pass
    return " ".join(map(str, filtered_quantities))

if __name__ == "__main__":
    n = int(input())
    inventory_input = [input() for _ in range(n)]
    threshold = int(input())
    output = filter_inventory_quantities(inventory_input, threshold)
    print(output)
```

***Status :*** Correct **Marks : 10/10**

3. Problem Statement

Gowshik is working on a task that involves taking two lists of integers as input, finding the element-wise sum of the corresponding elements, and then creating a tuple containing the sum values.

Write a program to help Gowshik with this task.

Example:

Given list:

[1, 2, 3, 4]

[3, 5, 2, 1]

An element-wise sum of the said tuples: (4, 7, 5, 5)

***Input Format***

The first line of input consists of a single integer n, representing the length of the input lists.

The second line of input consists of n integers separated by commas, representing the elements of the first list.

The third line of input consists of n integers separated by commas, representing the elements of the second list.

***Output Format***

The output is a single line containing a tuple of integers separated by commas, representing the element-wise sum of the corresponding elements from the two input lists.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 4
1, 2, 3, 4
3, 5, 2, 1
Output: (4, 7, 5, 5)

***Answer***

```
def element_wise_sum(list1, list2):
    """
```

```
    Calculates the element-wise sum of two lists of integers.

    Args:
        list1: The first list of integers.
        list2: The second list of integers.

    Returns:
        A tuple containing the element-wise sums.  Returns an empty tuple if
        either input list is empty or their lengths don't match.
    """
    if not list1 or not list2 or len(list1) != len(list2):
        return ()  # Return an empty tuple for invalid input

    sum_list = [list1[i] + list2[i] for i in range(len(list1))]
    return tuple(sum_list)

if __name__ == "__main__":
    n = int(input())
    input1 = input().split(',')
    list1 = [int(x.strip()) for x in input1]
    input2 = input().split(',')
    list2 = [int(x.strip()) for x in input2]

    result_tuple = element_wise_sum(list1, list2)
    print(result_tuple)
```

***Status :*** Correct **Marks : 10/10**

4. Problem Statement

Ella is analyzing the sales data for a new online shopping platform. She has a record of customer transactions where each customer's data includes their ID and a list of amounts spent on different items. Ella needs to determine the total amount spent by each customer and identify the highest single expenditure for each customer.

Your task is to write a program that computes these details and displays them in a dictionary.

***Input Format***

The first line of input consists of an integer n, representing the number of customers.

Each of the next n lines contains a numerical customer ID followed by integers representing the amounts spent on different items.

***Output Format***

The output displays a dictionary where the keys are customer IDs and the values are lists containing two integers: the total expenditure and the maximum single expenditure.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 2
101 100 150 200
102 50 75 100
Output: {101: [450, 200], 102: [225, 100]}

***Answer***

```
def analyze_customer_transactions(transactions):
    """
    Analyzes customer transactions to calculate total expenditure and maximum single expenditure for each customer.

    Args:
        transactions: A list of strings, where each string represents a customer transaction.
                      Each string starts with the customer ID, followed by the amounts spent on items.

    Returns:
        A dictionary where keys are customer IDs and values are lists containing
        [total_expenditure, maximum_single_expenditure].
    """
    customer_data = {}
    for transaction in transactions:
        data = list(map(int, transaction.split()))
        customer_id = data[0]
```

```
        amounts = data[1:]
        total_expenditure = sum(amounts)
        maximum_single_expenditure = max(amounts) if amounts else 0  # Handle
empty amounts list
        customer_data[customer_id] = [total_expenditure,
maximum_single_expenditure]
    return customer_data

if __name__ == "__main__":
    n = int(input())
    transactions = [input() for _ in range(n)]
    result = analyze_customer_transactions(transactions)
    print(result)
```

*Status :* Correct *Marks : 10/10*

5. Problem Statement

Professor Adams needs to analyze student participation in three recent academic workshops. She has three sets of student IDs: the first set contains students who registered for the workshops, the second set contains students who actually attended, and the third set contains students who dropped out.

Professor Adams needs to determine which students who registered also attended, and then identify which of these students did not drop out.

Help Professor Adams identify the students who registered, attended, and did not drop out of the workshops.

***Input Format***

The first line of input consists of integers, representing the student IDs who registered for the workshops.

The second line consists of integers, representing the student IDs who attended the workshops.

The third line consists of integers, representing the student IDs who dropped out of the workshops.

***Output Format***

The first line of output displays the intersection of the first two sets, which shows the IDs of students who registered and attended.

The second line displays the result after removing student IDs that are in the third set (dropped out), showing the IDs of students who both attended and did not drop out.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 1 2 3
2 3 4
3 4 5
Output: {2, 3}
{2}

***Answer***

```
def analyze_workshop_participation(registered_str, attended_str, dropped_out_str):
    """
    Identifies students who registered, attended, and did not drop out of workshops.

    Args:
        registered_str: A string of space-separated integers representing registered student IDs.
        attended_str: A string of space-separated integers representing attended student IDs.
        dropped_out_str: A string of space-separated integers representing dropped-out student IDs.

    Returns:
        A tuple containing two sets:
        - The set of students who registered and attended.
        - The set of students who registered, attended, and did not drop out.
    """
    registered = set(map(int, registered_str.split()))
```

```
    attended = set(map(int, attended_str.split()))
    dropped_out = set(map(int, dropped_out_str.split()))

    registered_and_attended = registered.intersection(attended)
    attended_and_did_not_drop_out =
registered_and_attended.difference(dropped_out)

    return registered_and_attended, attended_and_did_not_drop_out

if __name__ == "__main__":
    registered_input = input()
    attended_input = input()
    dropped_out_input = input()

    registered_attended_set, attended_not_dropped_out_set =
analyze_workshop_participation(
        registered_input, attended_input, dropped_out_input
    )

    print(registered_attended_set)
    print(attended_not_dropped_out_set)
```

***Status :*** Correct **Marks : 10/10**

# Rajalakshmi Engineering College

Name: gladwin antony.A
Email: 241501057@rajalakshmi.edu.in
Roll no: 241501057
Phone: 8015799480
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 5_PAH

Attempt : 1
Total Mark : 60
Marks Obtained : 56

## Section 1 : Coding

1. Problem Statement

Sophia is organizing a list of event IDs representing consecutive days of an event. She needs to group these IDs into consecutive sequences. For example, if the IDs 3, 4, and 5 appear consecutively, they should be grouped.

Write a program that helps Sophia by reading the total number of event IDs and the IDs themselves, then display each group of consecutive IDs in tuple format.

***Input Format***

The first line of input consists of an integer n, representing the number of event IDs.

The next n lines contain integers representing the event IDs, where each integer corresponds to an event ID.

***Output Format***

The output should display each group of consecutive event IDs in a tuple format. Each group should be printed on a new line, and single event IDs should be displayed as a single-element tuple.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 3
1
2
3
Output: (1, 2, 3)

***Answer***

```
# Read the total number of event IDs
n = int(input())

# Read the event IDs into a list
event_ids = []
for _ in range(n):
    event_ids.append(int(input()))

# Initialize a list to store the groups of consecutive IDs
groups = []

# If there are no event IDs, there's nothing to process
if not event_ids:
    pass # Do nothing if the list is empty
else:
    # Initialize the current group with the first event ID
    current_group = [event_ids[0]]

    # Iterate through the event IDs starting from the second one
    for i in range(1, n):
        # Check if the current event ID is consecutive to the last ID in the current
group
```

```
        if event_ids[i] == current_group[-1] + 1:
            # If it's consecutive, add it to the current group
            current_group.append(event_ids[i])
        else:
            # If it's not consecutive, the current group has ended
            # Add the completed current group (as a tuple) to the list of groups
            groups.append(tuple(current_group))
            # Start a new current group with the non-consecutive event ID
            current_group = [event_ids[i]]

    # After the loop, add the last current_group to the list of groups
    # This handles the very last sequence of consecutive IDs
    groups.append(tuple(current_group))

# Print each group on a new line, formatted as a tuple
for group in groups:
    print(group)
```

***Status :*** Partially correct **Marks : 6/10**

2. Problem Statement

Mia is organizing a list of integers into a series of pairs for his new project. She wants to create pairs of consecutive integers from the list. The last integer should be paired with None to complete the series. The pairing happens as follows: ((Element 1, Element 2), (Element 2, Element 3)..... (Element n, None)).

Your task is to help Henry by writing a Python program that reads a list of integers, forms these pairs, and displays the result in tuple format.

***Input Format***

The first line of input consists of an integer n, representing the number of elements in the tuple.

The second line of input contains n space-separated integers, representing the elements of the tuple.

***Output Format***

The output displays a tuple containing pairs of consecutive integers from the

input. The last integer in the tuple is paired with 'None'.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 3
5 10 15
Output: ((5, 10), (10, 15), (15, None))

***Answer***

```
# Read the number of elements in the list
n = int(input())

# Read the space-separated integers and convert them to a list of integers
elements = list(map(int, input().split()))

# Initialize an empty list to store the generated pairs
pairs_list = []

# Iterate through the list to create pairs
# The loop runs from the first element up to the second-to-last element
for i in range(n - 1):
    # Create a tuple of the current element and the next element
    pair = (elements[i], elements[i+1])
    # Add the created pair to the list of pairs
    pairs_list.append(pair)

# After the loop, handle the last element
# The last element of the original list is paired with None
last_pair = (elements[n-1], None)
# Add this special last pair to the list
pairs_list.append(last_pair)

# Convert the list of pairs into a tuple of tuples
# This is done by casting the list to a tuple
result_tuple = tuple(pairs_list)

# Print the final result in the specified tuple format
print(result_tuple)
```

***Status :*** Correct **Marks : 10/10**

3. Problem Statement

Tom wants to create a dictionary that lists the first n prime numbers, where each key represents the position of the prime number, and the value is the prime number itself.

Help Tom generate this dictionary based on the input she provides.

***Input Format***

The input consists of an integer n, representing the number of prime numbers Tom wants to generate.

***Output Format***

The output displays the generated dictionary where each key is an integer from 1 to n, and the corresponding value is the prime number.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 4
Output: {1: 2, 2: 3, 3: 5, 4: 7}

***Answer***

```
def is_prime(num):
    """
    Checks if a given number is prime.

    Args:
        num (int): The number to check.

    Returns:
        bool: True if the number is prime, False otherwise.
    """
    if num < 2:
        return False
```

```
    # Iterate from 2 up to the square root of num
    # If num is divisible by any number in this range, it's not prime
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

# Read the input integer n, representing the number of prime numbers to
generate
n = int(input())

# Initialize an empty dictionary to store the prime numbers
prime_dict = {}
# Initialize a counter for the position of the prime number (1-indexed)
prime_position = 1
# Start checking for prime numbers from 2
current_number = 2

# Loop until we have found 'n' prime numbers
while prime_position <= n:
    # Check if the current_number is prime
    if is_prime(current_number):
        # If it's prime, add it to the dictionary
        # The key is its position, and the value is the prime number itself
        prime_dict[prime_position] = current_number
        # Increment the prime position counter
        prime_position += 1
    # Move to the next number to check
    current_number += 1

# Print the generated dictionary
print(prime_dict)
```

***Status :*** Correct **Marks : 10/10**

4. Problem Statement

Maya wants to create a dictionary that maps each integer from 1 to a given number n to its square. She will use this dictionary to quickly reference the square of any number up to n.

Help Maya generate this dictionary based on the input she provides.

***Input Format***

The input consists of an integer n, representing the highest number for which Maya wants to calculate the square.

***Output Format***

The output displays the generated dictionary where each key is an integer from 1 to n, and the corresponding value is its square.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 5
Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

***Answer***

```
# Read the input integer n, representing the highest number for which to calculate the square
n = int(input())

# Initialize an empty dictionary to store the numbers and their squares
squares_dict = {}

# Iterate from 1 up to n (inclusive)
for i in range(1, n + 1):
    # Calculate the square of the current number
    square = i * i
    # Add the number as the key and its square as the value to the dictionary
    squares_dict[i] = square

# Print the generated dictionary
print(squares_dict)
```

***Status :*** Correct ***Marks : 10/10***

5. Problem Statement

Jordan is creating a program to process a list of integers. The program should take a list of integers as input, remove any duplicate integers while preserving their original order, concatenate the remaining unique integers into a single string, and then print the result.

Help Jordan in implementing the same.

***Input Format***

The input consists of space-separated integers representing the elements of the set.

***Output Format***

The output prints a single integer formed by concatenating the unique integers from the input in the order they appeared.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 11 11 33 50

Output: 113350

***Answer***

```
# Read the input as a string of space-separated integers
input_str = input()

# Split the string by spaces to get individual string representations of numbers
str_elements = input_str.split()

# Initialize an empty list to store unique integers in their original order
unique_elements = []

# Initialize a set to keep track of elements that have already been added to unique_elements
# Using a set allows for efficient O(1) lookup for checking duplicates
seen_elements = set()
```

```
# Iterate through each string element from the input
for s_elem in str_elements:
    # Convert the string element to an integer
    int_elem = int(s_elem)

    # Check if the integer element has not been seen before
    if int_elem not in seen_elements:
        # If it's a new unique element, add it to the unique_elements list
        unique_elements.append(int_elem)
        # Also add it to the seen_elements set to mark it as seen
        seen_elements.add(int_elem)

# Initialize an empty list to store the string representations of unique integers
result_str_parts = []

# Iterate through the list of unique integers
for u_elem in unique_elements:
    # Convert each unique integer back to its string representation
    result_str_parts.append(str(u_elem))

# Concatenate all the string parts into a single string
final_result_string = "".join(result_str_parts)

# Print the final concatenated string
print(final_result_string)
```

***Status :*** Correct **Marks : 10/10**

6. Problem Statement

Rishi is working on a program to manipulate a set of integers. The program should allow users to perform the following operations:

Find the maximum value in the set.Find the minimum value in the set.Remove a specific number from the set.

The program should handle these operations based on user input. If the user inputs an invalid operation choice, the program should indicate that the choice is invalid.

***Input Format***

The first line contains space-separated integers that will form the initial set. Each integer x is separated by a space.

The second line contains an integer ch, representing the user's choice:

- 1 to find the maximum value
- 2 to find the minimum value
- 3 to remove a specific number from the set

If ch is 3, the third line contains an integer n1, which is the number to be removed from the set.

***Output Format***

The first line of output prints the original set in descending order.

For choice 1: Print the maximum value from the set.

For choice 2: Print the minimum value from the set.

For choice 3: Print the set after removing the specified number, in descending order.

For invalid choices: Print "Invalid choice".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 1 2 3 4 5
1
Output: {5, 4, 3, 2, 1}
5

***Answer***

```
# Read the initial set elements as a space-separated string
input_elements_str = input()

# Convert the string elements to integers and then to a set
```

```
# Using a set automatically handles duplicates and provides efficient operations
initial_set = set(map(int, input_elements_str.split()))

# Convert the set to a list, sort it in descending order, and format for printing
# This step is done to ensure the initial set is printed in descending order as
required
sorted_initial_list = sorted(list(initial_set), reverse=True)
# Manually format the output string to match the desired set representation {x, y,
z}
initial_set_output_str = "{" + ", ".join(map(str, sorted_initial_list)) + "}"
print(initial_set_output_str)

# Read the user's choice for the operation
choice = int(input())

# Perform operations based on the user's choice
if choice == 1:
    # Choice 1: Find and print the maximum value in the set
    if initial_set: # Ensure the set is not empty before finding max
        print(max(initial_set))
    else:
        # This case might not be hit based on constraints, but good practice
        print("Set is empty, no maximum value.")
elif choice == 2:
    # Choice 2: Find and print the minimum value in the set
    if initial_set: # Ensure the set is not empty before finding min
        print(min(initial_set))
    else:
        # This case might not be hit based on constraints, but good practice
        print("Set is empty, no minimum value.")
elif choice == 3:
    # Choice 3: Remove a specific number from the set
    # Read the number to be removed
    num_to_remove = int(input())

    # Use discard() to remove the element. discard() does not raise an error
    # if the element is not found in the set, which is safer than remove().
    initial_set.discard(num_to_remove)

    # Convert the modified set to a list, sort it in descending order, and format for
printing
    sorted_modified_list = sorted(list(initial_set), reverse=True)
```

```
    # Manually format the output string to match the desired set representation {x,
y, z}
    modified_set_output_str = "{" + ", ".join(map(str, sorted_modified_list)) + "}"
    print(modified_set_output_str)
else:
    # Invalid choice: Print an error message
    print("Invalid choice")
```

***Status :*** Correct ***Marks : 10/10***

# Rajalakshmi Engineering College

Name: gladwin antony.A
Email: 241501057@rajalakshmi.edu.in
Roll no: 241501057
Phone: 8015799480
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## NeoColab_REC_CS23221_Python Programming

## REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 39

## Section 1 : Coding

1. Problem Statement

Alex is tasked with managing the membership lists of several exclusive clubs. Each club has its own list of members, and Alex needs to determine the unique members who are part of exactly one club when considering all clubs together.

Your goal is to help Alex by writing a program that calculates the symmetric difference of membership lists from multiple clubs and then finds the total number of unique members.

***Input Format***

The first line of input consists of an integer k, representing the number of clubs.

The next k lines each contain a space-separated list of integers, where each

integer represents a member's ID.

***Output Format***

The first line of output displays the symmetric difference of the membership lists as a set.

The second line displays the sum of the elements in this symmetric difference.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 3
1 2 3
2 3 4
5 6 7
Output: {1, 4, 5, 6, 7}
23

***Answer***

```
# Read the number of clubs (k)
k = int(input())

# Initialize a list to store all the membership sets
all_membership_sets = []

# Read k lines of member IDs and convert each into a set
for _ in range(k):
    # Read a line of space-separated integers
    line_of_members = input().split()
    # Convert string elements to integers and then to a set
    membership_set = set(map(int, line_of_members))
    # Add the created set to our list of all membership sets
    all_membership_sets.append(membership_set)

# Calculate the symmetric difference of all sets
# Initialize the symmetric difference with the first set
if all_membership_sets:
    symmetric_diff_result = all_membership_sets[0]
    # Iterate through the rest of the sets, applying symmetric difference
cumulatively
```

```
    for i in range(1, k):
        symmetric_diff_result =
symmetric_diff_result.symmetric_difference(all_membership_sets[i])
else:
    # If there are no sets (though constraints say k >= 3, it's good practice)
    symmetric_diff_result = set()

# Print the symmetric difference set
# The output format requires a set, so direct printing of the set is appropriate
print(symmetric_diff_result)

# Calculate the sum of the elements in the symmetric difference set
sum_of_symmetric_diff = sum(symmetric_diff_result)

# Print the sum
print(sum_of_symmetric_diff)
```

***Status :*** Partially correct ***Marks : 9/10***

2. Problem Statement

Samantha is working on a text analysis tool that compares two words to find common and unique letters. She wants a program that reads two words, w1, and w2, and performs the following operations:

Print the letters common to both words, in alphabetical order.Print the letters that are unique to each word, in alphabetical order.Determine if the set of letters in the first word is a superset of the letters in the second word.Check if there are no common letters between the two words and print the result as a Boolean value.

Ensure the program ignores case differences and leading/trailing spaces in the input words.

Your task is to help Samantha in implementing the same.

***Input Format***

The first line of input consists of a string representing the first word, w1.

The second line consists of a string representing the second word, w2.

***Output Format***

The first line of output should display the sorted letters common to both words, printed as a list.

The second line should display the sorted letters that are unique to each word, printed as a list.

The third line should display a Boolean value indicating if the set of letters in w1 is a superset of the set of letters in w2.

The fourth line should display a Boolean value indicating if there are no common letters between w1 and w2.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: program
Peace

Output: ['a', 'p']
['c', 'e', 'g', 'm', 'o', 'r']
False
False

***Answer***

```
def analyze_words(w1, w2):
    # Clean up the input words: remove leading/trailing spaces and convert to lowercase
    w1 = w1.strip().lower()
    w2 = w2.strip().lower()

    # Convert words into sets of characters
    set_w1 = set(w1)
    set_w2 = set(w2)

    # Common letters between w1 and w2
    common_letters = sorted(set_w1 & set_w2)

    # Unique letters for each word (letters that are in w1 but not in w2 and vice versa)
```

```
    unique_letters = sorted((set_w1 - set_w2) | (set_w2 - set_w1))

    # Check if w1 is a superset of w2
    is_superset = set_w1 >= set_w2

    # Check if there are no common letters
    no_common_letters = len(common_letters) == 0

    # Output the results
    print(common_letters)
    print(unique_letters)
    print(is_superset)
    print(no_common_letters)

# Input: read the two words
w1 = input()
w2 = input()

# Call the function
analyze_words(w1, w2)
```

***Status :*** Correct **Marks : 10/10**

3. Problem Statement

Alex is working with grayscale pixel intensities from an old photo that has been scanned in a single row. To detect edges in the image, Alex needs to calculate the differences between each pair of consecutive pixel intensities.

Your task is to write a program that performs this calculation and returns the result as a tuple of differences.

***Input Format***

The first line of input contains an integer n, representing the number of pixel intensities.

The second line contains n space-separated integers representing the pixel intensities.

***Output Format***

The output displays a tuple containing the absolute differences between consecutive pixel intensities.

Refer to the sample output for format specifications.

***Sample Test Case***

Input: 5
200 100 20 80 10
Output: (100, 80, 60, 70)

***Answer***

```
def calculate_pixel_differences(n, pixel_intensities):
    # Create a list to store the absolute differences between consecutive pixel
intensities
    differences = []

    # Iterate over the pixel intensities and calculate the absolute differences
    for i in range(1, n):
        diff = abs(pixel_intensities[i] - pixel_intensities[i - 1])
        differences.append(diff)

    # Convert the list of differences to a tuple and return it
    return tuple(differences)

# Input: Read the number of pixel intensities
n = int(input())

# Input: Read the pixel intensities as a list of integers
pixel_intensities = list(map(int, input().split()))

# Call the function and print the result
result = calculate_pixel_differences(n, pixel_intensities)
print(result)
```

***Status :*** Correct ***Marks : 10/10***

4. Problem Statement

Noah, a global analyst at a demographic research firm, has been tasked with identifying which country experienced the largest population growth over a two-year period. He has a dataset where each entry consists of a country code and its population figures for two consecutive years. Noah needs to determine which country had the highest increase in population and present the result in a specific format.

Help Noah by writing a program that outputs the country code with the largest population increase, along with the increase itself.

***Input Format***

The first line of input consists of an integer N, representing the number of countries.

Each of the following N blocks contains three lines:

1. The first line is a country code.
2. The second line is an integer representing the population of the country in the first year.
3. The third line is an integer representing the population of the country in the second year.

***Output Format***

The output displays the country code and the population increase in the format {code: difference}, where code is the country code and difference is the increase in population.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 3
01
1000
1500
02

2000
2430
03
1500
3000

Output: {03:1500}

***Answer***

```
def find_largest_population_increase(n):
    max_increase = 0
    country_code = ""

    for _ in range(n):
        # Read the country code
        code = input().strip()

        # Read the populations for both years
        pop1 = int(input().strip())
        pop2 = int(input().strip())

        # Calculate the population increase
        increase = pop2 - pop1

        # Update if this country has the largest increase
        if increase > max_increase:
            max_increase = increase
            country_code = code

    # Output the result in the required format
    print(f"{{{country_code}:{max_increase}}}")

# Input: Read the number of countries
N = int(input().strip())

# Call the function to process the data and print the result
find_largest_population_increase(N)
```

***Status :*** Correct ***Marks : 10/10***