Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
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 : 17

Section 1 : MCQ

1. 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: 'D', 3: 'C'}

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

2. 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**

3. 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**

4. 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**

5. 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

TrueTrue

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

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

```
set1 = {1, 2, 3}
```

```
set2 = set1.copy()
set2.add(4)
print(set1)
```

Answer

{1,2,3,4}

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

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

Answer

False

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

8. 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) t1=t+(3,4)2) max(t)

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

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

Answer

( )

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

10. What will be the output?

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

Answer

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

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

11. Which of the following is a Python tuple?

Answer

(1, 2, 3)

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

12. 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***

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

Answer

t[3] = 45

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

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

Answer

Keys must be integers

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

15. 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***

16. 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***

17. 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***

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

Answer

Values of a dictionary must be unique

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

19. 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***

20. 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***

Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
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

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

```
# Function to analyze student participation
def analyze_participation(registered, attended, dropped_out):
    # Convert input lists to sets for easier set operations
    registered_set = set(registered)
    attended_set = set(attended)
    dropped_out_set = set(dropped_out)

    # Find students who registered and attended
    registered_and_attended = registered_set.intersection(attended_set)

    # Find students who attended and did not drop out
    attended_and_not_dropped =
registered_and_attended.difference(dropped_out_set)
```

```
    return registered_and_attended, attended_and_not_dropped

# Input reading
registered = list(map(int, input().split()))
attended = list(map(int, input().split()))
dropped_out = list(map(int, input().split()))

# Get the results
registered_and_attended, attended_and_not_dropped =
analyze_participation(registered, attended, dropped_out)

# Print the outputs in the required format
print(registered_and_attended)
print(attended_and_not_dropped)
```

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

2. 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

```
# Function to calculate the element-wise sum of two lists
def elementwise_sum(n, list1, list2):
    # Calculate the sum of corresponding elements
    result = tuple(a + b for a, b in zip(list1, list2))
    return result

# Input reading
n = int(input())
list1 = list(map(int, input().split(',')))
list2 = list(map(int, input().split(',')))

# Get the result
result = elementwise_sum(n, list1, list2)

# Print the output in the required format
print(result)
```

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

3. 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

```
# You are using Python
# Function to analyze product ID frequencies
def analyze_product_ids(n, product_ids):
    frequency_dict = {}

    # Count the frequency of each product ID
    for product_id in product_ids:
        if product_id in frequency_dict:
            frequency_dict[product_id] += 1
        else:
            frequency_dict[product_id] = 1

    # Calculate total unique product IDs
    total_unique_ids = len(frequency_dict)

    # Calculate average frequency
    total_frequency = sum(frequency_dict.values())
    average_frequency = total_frequency / total_unique_ids if total_unique_ids > 0
else 0

    return frequency_dict, total_unique_ids, average_frequency

# Input reading
n = int(input())
product_ids = [int(input()) for _ in range(n)]

# Get the results
frequency_dict, total_unique_ids, average_frequency = analyze_product_ids(n,
product_ids)

# Print the outputs in the required format
print(frequency_dict)
```

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

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

4. 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

```
# You are using Python
# Function to filter quantities based on the threshold
def filter_quantities(inventory, threshold):
    filtered_quantities = []

    for item_id, quantities in inventory:
        # Use filter to get quantities greater than the threshold
        filtered = list(filter(lambda x: x > threshold, quantities))
        filtered_quantities.extend(filtered)

    return filtered_quantities

# Input reading
N = int(input())
inventory = []

for _ in range(N):
    item = eval(input().strip())  # Read the tuple input
    inventory.append(item)

threshold = int(input())

# Get the filtered quantities
result = filter_quantities(inventory, threshold)

# Print the output in the required format
print(" ".join(map(str, result)))
```

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

5. 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

```
# You are using Python
# Function to analyze customer expenditures
def analyze_expenditures(n, customer_data):
    expenditure_dict = {}

    for data in customer_data:
        # Split the data into customer ID and amounts
        amounts = list(map(int, data.split()))
        customer_id = amounts[0]
        expenditures = amounts[1:]
```

```
        # Calculate total expenditure and maximum single expenditure
        total_expenditure = sum(expenditures)
        max_expenditure = max(expenditures)

        # Store the results in the dictionary
        expenditure_dict[customer_id] = [total_expenditure, max_expenditure]

    return expenditure_dict

# Input reading
n = int(input())
customer_data = [input().strip() for _ in range(n)]

# Get the results
result = analyze_expenditures(n, customer_data)

# Print the output in the required format
print(result)
```

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

Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
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 : 60

Section 1 : Coding

1. 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

```
# You are using Python
def is_prime(num):
    """Check if a number is prime."""
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

def generate_primes(n):
    """Generate a dictionary of the first n prime numbers."""
    primes = {}
    count = 0
    num = 2  # Start checking for prime numbers from 2

    while count < n:
        if is_prime(num):
            count += 1
            primes[count] = num
        num += 1

    return primes

# Input reading
n = int(input())

# Generate the dictionary of prime numbers
result = generate_primes(n)

# Print the output in the required format
print(result)
```

Status : Correct *Marks : 10/10*

2. 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

```
def main():
    # Input reading
    initial_set = set(map(int, input().strip().split()))

    # Display the original set in descending order
    print(f"{{{', '.join(map(str, sorted(initial_set, reverse=True)))}}}")

    # User choice input
    ch = int(input().strip())

    if ch == 1:
        # Find the maximum value
        max_value = max(initial_set)
        print(max_value)
    elif ch == 2:
        # Find the minimum value
        min_value = min(initial_set)
        print(min_value)
    elif ch == 3:
        # Remove a specific number from the set
        n1 = int(input().strip())
        if n1 in initial_set:
            initial_set.remove(n1)
        # Display the set after removal in descending order
        print(f"{{{', '.join(map(str, sorted(initial_set, reverse=True)))}}}")
    else:
        # Invalid choice
        print("Invalid choice")

# Run the program
main()
```

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

3. 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

```
def group_consecutive_event_ids(n, event_ids):
```

```
    # Initialize an empty list to store the result groups
    result = []

    # Start iterating through the event_ids
    current_group = [event_ids[0]]

    for i in range(1, n):
        # If the current event_id is consecutive to the last one, add it to the current
group
        if event_ids[i] == event_ids[i - 1] + 1:
            current_group.append(event_ids[i])
        else:
            # If it's not consecutive, finalize the current group and start a new one
            result.append(tuple(current_group))
            current_group = [event_ids[i]]

    # Don't forget to add the last group
    result.append(tuple(current_group))

    # Output all the groups
    for group in result:
        if len(group) == 1:
            # Special case: single-element tuples should be printed without the
comma
            print(f"({group[0]})")
        else:
            print(group)

# Read input
n = int(input())  # number of event IDs
event_ids = [int(input()) for _ in range(n)]  # list of event IDs

# Call the function to group and print the results
group_consecutive_event_ids(n, event_ids)
```

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

4. 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

```
# You are using Python
def create_consecutive_pairs(n, integers):
    # List to hold the consecutive pairs
    pairs = []

    # Loop through the integers to create consecutive pairs
    for i in range(n - 1):
        pairs.append((integers[i], integers[i + 1]))

    # Pair the last element with None
    pairs.append((integers[-1], None))
```

```
    # Output the pairs as a tuple
    print(tuple(pairs))

# Read input
n = int(input())  # Number of elements in the list
integers = list(map(int, input().split()))  # List of integers

# Call the function to generate and print the pairs
create_consecutive_pairs(n, integers)
```

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

5. 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

```
# You are using Python
def generate_square_dict(n):
```

```
    # Create a dictionary using a dictionary comprehension
    square_dict = {i: i**2 for i in range(1, n+1)}
    # Print the generated dictionary
    print(square_dict)

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

# Generate and display the dictionary
generate_square_dict(n)
```

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

6. 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

```
# You are using Python
def process_integers(input_list):
    seen = set()  # To track unique integers
    unique_integers = []  # To store the unique integers in order

    # Iterate over the input list
    for num in input_list:
        if num not in seen:
            unique_integers.append(str(num))  # Convert to string for concatenation
            seen.add(num)

    # Join the list of strings into a single string and print the result
    print(''.join(unique_integers))

# Read input
input_list = list(map(int, input().split()))

# Call the function to process the input and print the result
process_integers(input_list)
```

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

Rajalakshmi Engineering College

Name: Prithvi Yogeswaraa M
Email: 241501154@rajalakshmi.edu.in
Roll no: 2116241501154
Phone: 9345507776
Branch: REC
Department: I AIML AD
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 : 37.5

Section 1 : Coding

1. Problem Statement

Riley is analyzing DNA sequences and needs to determine which bases match at the same positions in two given DNA sequences. Each DNA sequence is represented as a tuple of integers, where each integer corresponds to a DNA base.

Your task is to write a program that compares these two sequences and identifies the bases that match at the same positions and print it.

Input Format

The first line of input consists of an integer n, representing the size of the first tuple.

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

The third line of input consists of an integer m, representing the size of the second tuple.

The fourth line contains m space-separated integers, representing the elements of the second DNA sequence tuple.

Output Format

The output is a space-separated integer of the matching bases at the same positions in both sequences.

Refer to the sample output for format specifications.

Sample Test Case

Input: 4
5 1 8 4
4
4 1 8 2

Output: 1 8

Answer

```
# You are using Python
def compare_dna_sequences():
    # Read input
    n = int(input())  # size of the first sequence
    sequence_1 = list(map(int, input().split()))  # first DNA sequence
    m = int(input())  # size of the second sequence
    sequence_2 = list(map(int, input().split()))  # second DNA sequence

    # Initialize a list to hold matching bases
    matching_bases = []

    # Find the smallest size to avoid index out of range errors
    min_length = min(n, m)

    # Compare the sequences and find matching bases at the same positions
    for i in range(min_length):
        if sequence_1[i] == sequence_2[i]:
            matching_bases.append(sequence_1[i])
```

```
    # Print the matching bases as space-separated integers
    print(" ".join(map(str, matching_bases)))

# Call the function to run the comparison
compare_dna_sequences()
```

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

2. Problem Statement

James is an engineer working on designing a new rocket propulsion system. He needs to solve a quadratic equation to determine the optimal launch trajectory. The equation is of the form ax2 +bx+c=0.

Your task is to help James find the roots of this quadratic equation. Depending on the discriminant, the roots might be real and distinct, real and equal, or complex. Implement a program to determine and display the roots of the equation based on the given coefficients.

Input Format

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

The second line contains three space-separated integers a,b, and c representing the coefficients of the quadratic equation.

Output Format

The output displays:

1. If the discriminant is positive, display the two real roots.
2. If the discriminant is zero, display the repeated real root.
3. If the discriminant is negative, display the complex roots as a tuple with real and imaginary parts.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 5 6
Output: (-2.0, -3.0)

Answer

```
# You are using Python
import cmath

def find_roots(a, b, c):
    # Calculate the discriminant
    discriminant = b**2 - 4*a*c

    # If discriminant is positive, two real roots
    if discriminant > 0:
        root1 = (-b + cmath.sqrt(discriminant)) / (2*a)
        root2 = (-b - cmath.sqrt(discriminant)) / (2*a)
        return (root1.real, root2.real)

    # If discriminant is zero, one real root (repeated)
    elif discriminant == 0:
        root = -b / (2*a)
        return (root,)

    # If discriminant is negative, two complex roots
    else:
        root1 = (-b + cmath.sqrt(discriminant)) / (2*a)
        root2 = (-b - cmath.sqrt(discriminant)) / (2*a)
        return (root1.real, root1.imag), (root2.real, root2.imag)

# Read input
N = int(input())  # This is always 3, representing the coefficients
a, b, c = map(int, input().split())  # Coefficients of the quadratic equation

# Get the roots based on the discriminant
roots = find_roots(a, b, c)

# Print the output in the required format
if len(roots) == 1:
    print(f"({roots[0]})")  # Single root for discriminant 0
else:
```

```
    # If complex roots, print as tuples
    if isinstance(roots[0], tuple):
        print(f"({roots[0]}, {roots[1]})")  # Complex roots
    else:
        print(f"({roots[0]}, {roots[1]})")  # Two real roots
```

Status : Partially correct **Marks : 7.5/10**

3. Problem Statement

Riya owns a store and keeps track of item prices from two different suppliers using two separate dictionaries. He wants to compare these prices to identify any differences. Your task is to write a program that calculates the absolute difference in prices for items that are present in both dictionaries. For items that are unique to one dictionary (i.e., not present in the other), include them in the output dictionary with their original prices.

Help Riya to implement the above task using a dictionary.

Input Format

The first line of input consists of an integer n1, representing the number of items in the first dictionary.

The next n1 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

The following line consists of an integer n2, representing the number of items in the second dictionary

The next n2 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

Output Format

The output should display a dictionary that includes:

1. For items common to both dictionaries, the absolute difference between their prices.
2. For items that are unique to one dictionary, the original price from that dictionary.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
4
4
1
8
7
Output: {4: 4, 8: 7}

Answer

```
# You are using Python
def compare_prices():
    # Read the number of items in the first dictionary
    n1 = int(input())  # Number of items in the first dictionary
    dict1 = {}

    # Read items and prices for the first dictionary
    for _ in range(n1):
        item = int(input())  # Item key
        price = int(input())  # Item price
        dict1[item] = price

    # Read the number of items in the second dictionary
    n2 = int(input())  # Number of items in the second dictionary
    dict2 = {}

    # Read items and prices for the second dictionary
    for _ in range(n2):
        item = int(input())  # Item key
        price = int(input())  # Item price
        dict2[item] = price
```

```
    # Result dictionary to store the differences or prices
    result = {}

    # Compare the dictionaries
    for item in dict1:
        if item in dict2:
            # Item is in both dictionaries, calculate absolute price difference
            result[item] = abs(dict1[item] - dict2[item])
        else:
            # Item is unique to dict1, keep its original price
            result[item] = dict1[item]

    for item in dict2:
        if item not in dict1:
            # Item is unique to dict2, keep its original price
            result[item] = dict2[item]

    # Print the result dictionary
    print(result)

# Call the function
compare_prices()
```

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

4. 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

```
# You are using Python
def calculate_differences():
    # Read the number of pixel intensities
    n = int(input())  # Number of pixel intensities

    # Read the pixel intensities as a list
    pixels = list(map(int, input().split()))

    # List to store absolute differences
    differences = []

    # Calculate the absolute differences between consecutive pixel intensities
    for i in range(1, n):
        diff = abs(pixels[i] - pixels[i - 1])  # Absolute difference between consecutive
pixels
        differences.append(diff)

    # Convert the differences list to a tuple and print the result
    print(tuple(differences))

# Call the function
calculate_differences()
```

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