**Problem Statement**

**Title: Intersection of Two Linked Lists**

**Problem Description:** Ella wants to find the common songs between two different playlists represented as linked lists. She needs to find the intersection of these lists. Given two singly linked lists, find if they intersect and print the intersecting node. If no intersection, print null.

**Input Format:**

* The first line contains an integer n, the number of elements in LinkedList A.
* The second line contains n space-separated elements of LinkedList A.
* The third line contains an integer k, the number of elements in LinkedList B.
* The fourth line contains k space-separated elements of LinkedList B.

**Output Format:** Print the intersection point of both the lists. If there is no intersection, print null.

**Constraints:**

* The number of nodes in each linked list is in the range [0, 1000].
* The elements of the linked lists are integers.

**Examples:**

**Example 1:**

Input:

5

1 2 3 4 5

4

6 7 3 4 5

Output:

3

**Example 2:**

Input:

3

1 2 3

3

4 5 6

Output:

null

**Test Cases:**

**Test Case 1:**

Input:

5

10 20 30 40 50

5

5 15 30 40 50

Output:

30

**Test Case 2:**

Input:

4

8 12 16 20

5

3 4 5 12 16 20

Output:

12

**Test Case 3:**

Input:

3

7 14 21

3

3 6 9

Output:

null

**Test Case 4:**

Input:

6

1 2 3 4 5 6

4

10 20 30 40

Output:

null

**Test Case 5:**

Input:

7

1 2 3 4 5 6 7

5

5 6 7 8 9

Output:

5

**Test Case 6:**

Input:

7

1 2 3 4 5 6 7

5

5 6 8 9 10

Output:

5

**Solution in Python**

Here is a Python solution that can help find the intersection point of two linked lists:

python

Copy code

class ListNode:

def \_\_init\_\_(self, x):

self.val = x

self.next = None

def get\_intersection\_node(headA, headB):

if not headA or not headB:

return None

# Get the lengths of both lists

lenA, lenB = 0, 0

currA, currB = headA, headB

while currA:

lenA += 1

currA = currA.next

while currB:

lenB += 1

currB = currB.next

# Align both lists to the same starting point

currA, currB = headA, headB

if lenA > lenB:

for \_ in range(lenA - lenB):

currA = currA.next

else:

for \_ in range(lenB - lenA):

currB = currB.next

# Find the intersection point

while currA and currB:

if currA == currB:

return currA.val

currA = currA.next

currB = currB.next

return None

# Helper function to create a linked list from a list

def create\_linked\_list(lst):

if not lst:

return None

head = ListNode(lst[0])

current = head

for value in lst[1:]:

current.next = ListNode(value)

current = current.next

return head

def main():

import sys

input = sys.stdin.read

data = input().strip().split()

n = int(data[0])

listA = list(map(int, data[1:n+1]))

k = int(data[n+1])

listB = list(map(int, data[n+2:]))

headA = create\_linked\_list(listA)

headB = create\_linked\_list(listB)

intersection = get\_intersection\_node(headA, headB)

if intersection is not None:

print(intersection)

else:

print("null")

if \_\_name\_\_ == "\_\_main\_\_":

main()

This solution reads the input values, creates the linked lists, and then finds the intersection point using the get\_intersection\_node function. The output is either the intersecting node value or null if there is no intersection.