QUESTION NO 1

PLUS ONE

You are given a **large integer** represented as an integer array `digits`, where each `digits[i]` is the `i^th` digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. Increment the large integer by one and return *the resulting array of digits*.

∴ The large integer does not contain any leading `0's`.

Example 1:

```
Input: digits = [1,2,3]

Output: [1,2,4]

Explanation: The array represents the integer 123.

Incrementing by one gives 123 + 1 = 124.

Thus, the result should be [1,2,4].
```

Example 2:

```
Input: digits = [4,3,2,1]

Output: [4,3,2,2]

Explanation: The array represents the integer 4321.

Incrementing by one gives 4321 + 1 = 4322.

Thus, the result should be [4,3,2,2].
```

Example 3:

```
Input: digits = [9]

Output: [1,0]

Explanation: The array represents the integer 9.

Incrementing by one gives 9 + 1 = 10.

Thus, the result should be [1,0].
```

Constraints:

- `1 <= digits.length <= 100`
- `0 <= digits[i] <= 9`
- `digits` does not contain any leading `0's`

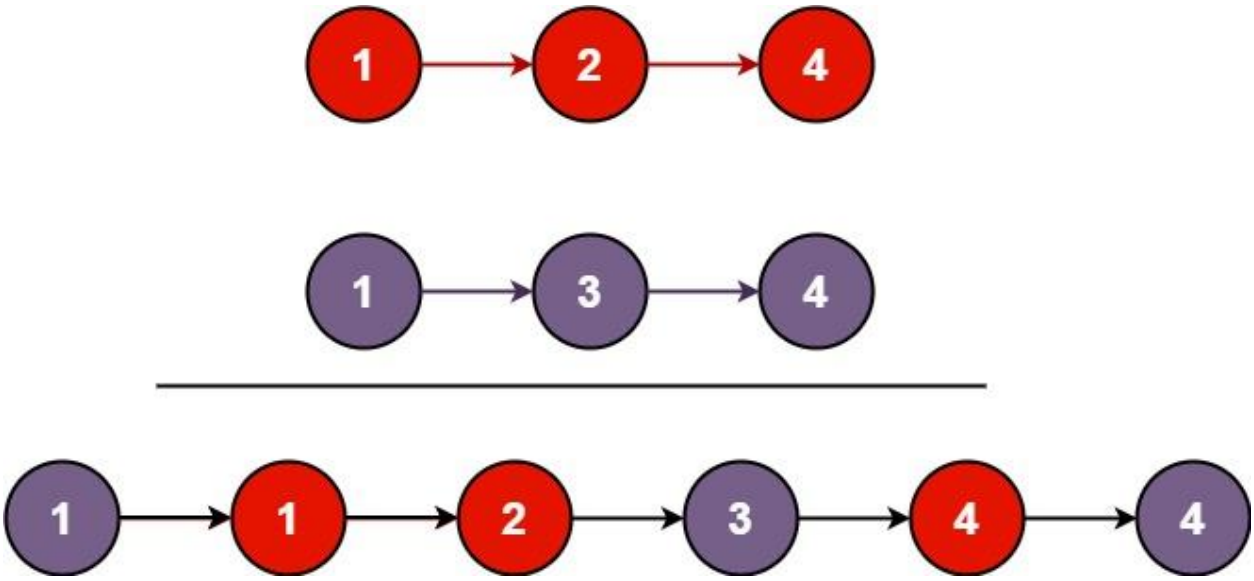
QUESTION NO 2

MERGE TWO SORTED LISTS

You are given the heads of two sorted linked lists `list1` and `list2`. Merge the two lists in a one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Your function should return `the head of merged linked list`.

Example 1:




```
Input: list1 = [1,2,4], list2 = [1,3,4]

Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []

Output: []
```

Example 3:

```
Input: list1 = [], list2 = [0]

Output: [0]
```

Constraints:

- The number of nodes in both lists is in the range `[0,50]`
- `-100 <= Node.val <= 100`
- Both `list1` and `list2` are sorted in **non-decreasing** order

QUESTION NO 3

PANGRAM

Given a string `sentence` containing only lowercase English letters, return `true` if `sentence` is a **pangram**, or `false` otherwise.

∴ A **pangram** is a sentence in which every letter of the English alphabet appears at least once.

Example 1:

```
Input: sentence = "thequickbrownfoxjumpsoverthelazydog"

Output: true

Explanation: sentence contains at least one of every letter of the English alphabet.
```

Example 2:

```
Input: sentence = "leetcode"

Output: false
```

Constraints:

- `1 <= sentence.length <= 1000`
- `sentence` consists of lowercase English letters only