# Code Challenge

Merge two linked lists.

## Specifications

- Read all of these instructions carefully. Name things exactly as described.
- Do all your work in a public repository called `data-structures-and-algorithms`, with a well-formatted, detailed top-level README.md.
- Create a new branch in your repo called `ll_merge`.
- Your top-level readme should contain a “Table of Contents” navigation to all of your challenges and implementations so far. (Don’t forget to update it!)
- This assignment should be completed within the `challenges` subdirectory of the repository.
- On your branch, create…
  - *C#*: a new .NET Core console project named `LLMerge`. Within your `Program.cs` create a new static method outside of `Main()` following the naming conventions below. Call your newly created method in `Main()` once complete.
  - *JavaScript*: a folder named `llMerge` which contains a file called `ll-merge.js`
  - *Python*: a folder named `ll_merge` which contains a file called `ll_merge.py`
  - *Java*: a static utility method `public static LinkedList mergeLists(LinkedList one, LinkedList two)` in your existing `LinkedList` class
- Include any language-specific configuration files required for this challenge to become an individual component, module, library, etc.
  - *NOTE: You can find an example of this configuration for your course in your class lecture repository.*

## Feature Tasks

- Write a function called `mergeLists` which takes two linked lists as arguments. Zip the two linked lists together into one so that the nodes alternate between the two lists and return a reference to the head of the zipped list. Try and keep additional space down to O(1). You have access to the Node class and all the properties on the Linked List class as well as the methods created in previous challenges.

## Example

`mergeLists(list1, list2)`

| Arg `list1` | Arg `list2` | Output |
|---|---|---|
| `head -> [1] -> [3] -> [2] -> X` | `head -> [5] -> [9] -> [4] -> X` | `head -> [1] -> [5] -> [3] -> [9] -> [2] -> [4] -> X` |
| `head -> [1] -> [3] -> X` | `head -> [5] -> [9] -> [4] -> X` | `head -> [1] -> [5] -> [3] -> [9] -> [4] -> X` |
| `head -> [1] -> [3] -> [2] -> X` | `head -> [5] -> [9] -> X` | `head -> [1] -> [5] -> [3] -> [9] -> [2] -> X` |

## Stretch Goal

Once you've achieved a working solution, implement another function that merges two sorted linked lists into a single sorted linked list.

## Requirements

Ensure your complete solution follows the standard requirements.

1. Write unit tests
2. Follow the template for a well-formatted README
3. Submit the assignment following these instructions