CS 245 - Lab 5 Fall, 2021

Lab 5 - Palindrome Linked List & Convert Binary to Integer In LL

The goal of this assignment is to demonstrate your mastery of linked lists. Code here.

Background

Part A: Create linked list

Linked lists are common data structures and the most asked interview questions. Create your own linked list data structure using your own custom `Node`. The data structure should have a `constructor` and the following functions, `add, reverse, and remove.`

Examples

```
Add
Add 5
Linked list: 1->2->3->4->NULL
result: 1->2->3->4->5->NULL

Add(with position)
Add 3 at position 2
Linked list: 1->2->4->NULL
result: 1->2->3->4->NULL (now 3 is at index 2, 4 was index 2 -> now 3)

Remove(with position)
Remove element at position 1
Linked list: 4->2->1->7->NULL
result: 4->1->7->NULL

Reverse
Linked list: 1->2->3->4->NULL
result: 4->3->2->1->NULL
```

Part B: Palindrome linked list

Palindrome is the sequence that reads the same backward as forward. Given the `head` of the singly linked list, determine the linked list whether it is palindrome or not. Return `true` if it is. Hints: you can use the reverse function.

- The value of each node is in range [1-9]
- Number of nodes in the list could be in range of [1, 10^5]

Examples

```
Input: head = [2,0,1,1,2,0]

Output: false

Input: head = [1,2,1]

Output: true

Input: head = [1,0]

Output: false

Input: head = [1]

Output: true
```

Part C: Convert Binary to Integer from LL

Given head which is a reference node to a singly-linked list. The value of each node in the linked list is either 0 or 1. The linked list holds the binary representation of a number.
Return the decimal value of the number in the linked list.

- The Linked List is not empty.
- Number of nodes will not exceed 30.
- Each node's value is either 0 or 1.

Here are some examples

```
Input: head = [1,0,1]
Output: 5
Explanation: (101) in binary is = 5

Input: head = [1]
Output: 1

Input: head = [0]
Output: 0

Input: head = [0,0]
Output: 0

Input: head = [1,0,0,1,0,0,1,1,1,0,0,0,0,0,0]
Output: 18880
```

Requirements

- You must submit a single Java file containing the class named **LinkedList**. You class **must contain the functions with the following headers(deduction for violation)**:

    - `public void add(int value)` **5%**

    This function should add a new node at the end of the linked list.

    - `public void add(int value, int position)` **5%**

    This function should add a new node at the given position(index).

    - `public Node remove(int position)` **10%**

    This function should both remove and return the node.

    - `public Node reverse(Node head)` **15%**

    This function should return the new head once the linked list is reversed.

    - `public boolean isPalindrome(Node head)` **20%**

    This function should return true if the linked list is palindrome, return false if not.

    - `public int getDecimalValue(Node head)` **20%**

    This function converts the binary number in a linked list to decimal.

Submission

- Download the source code for your implementation from Github Classroom. If you don't have a Github account, create one via github.com > Sign up.
- Select your name from the list of names available, which will link your GitHub ID. If you do not see your name on the list, speak with the instructor or one of the teaching assistants.
- Submit the repository link on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
    - 35% Part A
    - 20% Part B
    - 20% Part C

- 15% = Decomposition: in the eyes of the grader, the degree to which your solution follows the suggestions above or otherwise represents a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.