

Singly Linked List

****You do not need to write the Node class, createList()/printList()/count()/nodeAt() etc. functions unless specified in the problem.****

****Just design the function specified in the problem.****

****No need to write driver code.****

****This document is a compilation of previous semester's exam (midterm/final) problems and lab quiz problems. Try to practice relevant problems from online judges as well.****

1.

Question 2: CO3 [2 + 8 Points]

- I. **Write** down two disadvantages of Linked List over Array.
- II. You are given two linked lists of the same even length. Your task is to **complete** a method **pairwiseEqual()** that takes two singly linear linked list heads as arguments, checks if the linked lists are equal pairwise and returns True/False.
Two linked lists will be equal pairwise if the node values of every pair in Linked List 1 are equal to the node values of corresponding pair in Linked List 2 irrespective of their sequence [**i.e. the sequence does not matter**].
[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LISTS]

Sample Input	Returned Value	Explanation
head1 =10-->15-->34-->41 head2 = 15-->10-->34-->41	True	Linked List 1 Pairs: (10,15) ,(34,41) ; Linked List 2 Pairs: (15,10) , (34,41)
head1 =10-->15-->34-->42 head2 = 15-->10-->34-->41	False	Linked List 1 Pairs: (10,15) ,(34,42) ; Linked List 2 Pairs: (15,10) , (34,41)

Python Notation	Java Notation
def pairwiseEqual(h1, h2): # To Do	public boolean pairwiseEqual(Node h1, Node h2) { // To Do }

2.

You are given a non-dummy headed singly linear linked list containing positive integers. Your task is to complete the given method **IsSumPossible()** which takes the head of the linked list and an integer, *n* as input. If the integer *n* can be obtained by summing any two of the elements of the linked list, return True, else return False.

Page 1 of 3

Set-A

Hint: There might be more than one such pair possible, you don't have to check all such pairs.

[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LIST]

Sample Input	Returned Value	Explanation
list = 1→2→3→4→5 n = 7	True	Sum of the elements (3,4) or (2,5) equals 7
list = 1→2→4→5→6 n = 4	False	There are no two elements that make the sum 4. Note that though there is an element 4 itself in the list, the output will be False.
list = 5 n = 5	False	There is only one element so, the sum of two elements cannot be 5

Python Notation	Java Notation
<pre>def IsSumPossible(head, n): # To Do</pre>	<pre>public boolean IsSumPossible(Node head, int n) { // To Do }</pre>

3.

Suppose, you are working on a web application which has a customer support system. You are assigned to handle the customer support tickets. The support tickets are stored in a non-dummy headed singly linear linked list on a first-come first-served basis by IDs. However, due to a glitch, some tickets have been duplicated. Now, your task is to complete the given method **remove_Duplicates(head)** which takes the head of the linked list as input to remove the duplicate IDs ensuring that each ticket appears only once.

[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LIST]

Sample Input	Sample Output	Explanation
Input Tickets: 101 -> 103 -> 101 -> 102 -> 103 -> 104 -> 105 -> 105	Fixed Tickets: 101 -> 103 -> 102 -> 104 -> 105	All the duplicates of 101, 103 and 105 have been removed.
Input Tickets: 102 -> 101 -> 101 -> 102 -> 102 -> 102 -> 103 -> 104 -> 104	Fixed Tickets: 102 -> 101 -> 103 -> 104	All the duplicates of 102, 101 and 104 have been removed.

Python Notation	Java Notation
<pre>def remove_Duplicates(head): # To Do</pre>	<pre>public Node remove_Duplicates(Node head) { // To Do }</pre>

4.

You are given a non-dummy-headed, singly linear linked list containing positive integers. Your task is to complete the given method **reverseAndSwap()** that takes the **head** of the linked list and an **integer i** as input and returns the head of the modified list.

Your task is to

- Reverse the list from index 0 to i
- Swap the two parts of the list, i.e., the unchanged part from index i + 1 to total_nodes – 1 will come before the reversed part from index 0 to i in the new list. (where total_nodes = number of nodes in the linked list)

Check the given input and output for more clarification.

Note: You can assume that the index i will always be in the range 0 to total_nodes – 1. Assume Node class has elem and next variable. No need to write the Node class.

[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LIST, i.e. YOU CANNOT CREATE A NEW LINKED LIST]

Sample Input	Sample Output	Explanation
5 → 7 → 6 → 3 → 8 → 2 → 1 i = 3	8 → 2 → 1 → 3 → 6 → 7 → 5	The list is reversed from index 0 to 3 producing 3 → 6 → 7 → 5. The unchanged part (8 → 2 → 1) now sits before the reversed part, producing the output list.
5 → 7 → 6 → 3 → 8 → 2 → 1 i = 0	7 → 6 → 3 → 8 → 2 → 1 → 5	The list is reversed in index 0, producing 5. The unchanged part (7 → 6 → 3 → 8 → 2 → 1) now sits before the reversed part, producing the output list.
5 → 7 → 6 → 3 → 8 → 2 → 1 i = 6	1 → 2 → 8 → 3 → 6 → 7 → 5	The list is reversed from index 0 to 6, there is no part from the original list left.
5 → 7 → 6 → 3 → 8 → 2 → 1 i = 5	1 → 2 → 8 → 3 → 6 → 7 → 5	The list is reversed from index 0 to 5 producing 2 → 8 → 3 → 6 → 7 → 5. The only node remaining (Node 1) comes before the reversed part producing the output list.

Python Notation	Java Notation
<pre>def reverseAndSwap(head, i): # To Do</pre>	<pre>public Node reverseAndSwap(Node head, int i) { // To Do }</pre>

5.

You are given a Linked List, LL1, and an array, dist. Write a method **sum_dist(list, arr)** that takes a Linked List and an array as parameters. This method sums the node elements in the linked list that are away from the head by the elements in the array and returns the sum. Assume the Node class has only elem and next variable. **No need to write Node class and driver code.**

Sample Input	Sample Output	Explanation
LL1 = 10--> 16 --> -5 --> 9 --> 3 --> 4 dist = [2, 0, 5, 2, 8] Function Call: print(sum_dist(LL1, dist))	4	Node Element away from the head at distance 2 = -5 Node Element away from the head at distance 0 = 10 Node Element away from the head at distance 5 = 4 Node Element away from the head at distance 8 = Doesn't Exist, Considered as 0 The sum is: -5+10+4+-5+0 = 4

6.

ii) Suppose, you are in charge of collecting customer tokens in a line and have to store them in a singly linked list in a first-come, first-served order. Your company's policy is to serve senior citizens first, but you noticed they were standing towards the back, and you know their starting positions. Your task is to rearrange the tokens in a way so that senior citizens are moved to the front.

Implement the method named **rearrange_Tokens(head, seniorPos)**, which takes the head of the linked list and the starting position of the senior citizens. It rearranges senior citizens to the front and others to the back and returns the rearranged linked list's head. If the senior position is invalid, return the original list unchanged.

[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LIST]

Sample Input	Sample Output	Explanation
Tokens list: A3 → A9 → A4 → A2 → A7 → A8 → A1 Senior citizen position: 4	Rearranged Tokens list: A2 → A7 → A8 → A1 → A3 → A9 → A4	Here the senior citizens start from position 4, hence all the tokens from position 4 to the end of the list were brought to the front and others are behind that list.
Tokens list: A9 → A3 → A4 → A8 → A6 → A5 Senior citizen position: 5	Rearranged Tokens list: A6 → A5 → A9 → A3 → A4 → A8	Here the senior citizens start from position 5, hence all the tokens from position 5 to the end of the list were brought to the front and others are behind that list.

Python Notation	Java Notation
<pre>def rearrange_Tokens(head, seniorPos): # To Do</pre>	<pre>public Node rearrange_Tokens(Node head, int seniorPos) { // To Do }</pre>

7.

In a village, a wise elder kept a collection of stones, each marked with a number. These stones were arranged in ascending order, with the smallest stone at the beginning. One day, the elder asked you to help by collecting the first n smallest stones and calculate the product of the values to unlock a hidden power.

Your task is to help the elder by implementing the function `smallest_number_product(head, n)`. This function will receive two parameters:

- A sorted **linked list(head)**, where each node contains a number, **arranged in ascending order(smallest to largest)**.
- A number n , indicating how many of the smallest stones (nodes) you need to consider for multiplication.

****Your goal is to return the product of the first n smallest numbers from the sorted linked list.****

No other data structures can be used other than linked lists.

Consider that node class is already provided.

Sample Input	Sample Output	Explanation
1 -> 2 -> 3 -> 4 -> 5->6 n=4	Sample Output: Product= 24	$1*2*3*4= 24$
2-> 3 -> 9-> 20 -> 100 -> 110 n=3	Sample Output: Product= 54	$2*3*9= 54$

8.

In a village, a wise elder kept a collection of stones, each marked with a number. These stones were arranged in ascending order, with the smallest stone at the beginning. One day, the elder asked you to help by collecting the first n smallest stones and adding up their magical values to unlock a hidden power.

Your task is to help the elder by implementing the function `smallest_number_sum(head, n)`. This function will receive two parameters:

- A sorted **linked list**(`head`), where each node contains a number, **arranged in ascending order**(smallest to largest).
- A number n , indicating how many of the smallest stones you need to sum.

******Your goal is to return the sum of the first n smallest numbers from the sorted linked list.******

No other data structures can be used other than linked lists.

Consider that node class is already provided.

Sample Input	Sample Output	Explanation
1 -> 2 -> 3 -> 4 -> 5->6 n=4	Sample Output: Sum= 10	1+2+3+4= 10
2-> 3 -> 9-> 20 -> 100 -> 110 n=3	Sample Output: Sum=14	2+3+9= 14

9.

Question: Write a function that takes the **head of a Singly Linked List** and **adds corresponding values from both ends of the list (front value + end value)** and prints the subtraction of that sequence. The Linked List contains integer numbers as elements.

Notes:

1. For Simplicity, You can assume the Linked List has even number of elements
2. No need to write the Node class. Just assume Node class is there with two instance variables; **one is elem** and the **other one is next**.
3. You can use helper functions if necessary such as **length()** or **nodeAt()**.

Sample Given LinkedList	Sample Output 1	Explanation
1 → 2 → 3 → 4 → 11 → 9	-10	$(1 + 9) - (2 + 11) - (3 + 4) = -10$
Sample Given LinkedList	Sample Output 2	Explanation
1 → 2 → 3 → 1	-3	$(1 + 1) - (2 + 3) = -3$
Python Notation		Java Notation
<pre>def twoEndSum(head): # Your Code Here</pre>		<pre>static void twoEndSum(Node head){ // Your Code Here }</pre>

10.

Write a function that takes the **head of a Singly Linked List** and **subtracts corresponding values from both ends of the list (front value - end value)** and prints the summation of the sequence. The Linked List contains integer numbers as elements.

Notes:

1. For Simplicity, You can assume the Linked List has even number of elements
2. No need to write the Node class. Just assume Node class is there with two instance variables; **one is elem** and the **other one is next**.
3. You can use helper functions if necessary such as **length()** or **nodeAt()**.

Sample Input 1	Sample Output 1	Explanation
9 → 11 → 3 → 4 → 2 → 1	16	$(9 - 1) + (11 - 2) + (3 - 4) = 16$
Sample Input 2	Sample Output 2	Explanation
1 → 2 → 3 → 1	-1	$(1 - 1) + (2 - 3) = -1$
Python Notation		Java Notation
<pre>def twoEndSub(head): # Your Code Here</pre>		<pre>static void twoEndSub(Node head){ // Your Code Here }</pre>

11.

Write a function called **isPalindrome(head)**, that takes the head of a linked list in its parameter. The function does a very simple task. The function will return True if the linked list is actually a Palindrome otherwise it'll return False.

Sample Function Call	Sample Returned Result
head => 10 -> 43 -> 54 -> 43 -> 10 isPalindrome(head)	True
head => 1 -> 41 -> 4 -> 3 -> 10 isPalindrome(head)	False

Hint: You may want to write the **nodeAt()** method/function to use it in this task.

12.

Write a function called **revMatch(head1, head2)**, that takes two heads of two different linked lists in its parameter. The function checks whether the first linked list is matching with the second linked list in reverse order. If they match up then the function will return True otherwise False. You can assume that the length of these two linked lists are always equal.

Sample Function Call	Sample Returned Result
head1 => 1 -> 3 -> 5 -> 7 -> 4 head2 => 4 -> 7 -> 5 -> 3 -> 1 revMatch(head1, head2)	True
head1 => 1 -> 9 -> 2 -> 7 -> 4 head2 => 4 -> 7 -> 5 -> 3 -> 1 revMatch(head1, head2)	False

Hint: You may want to write the **nodeAt()** method/function to use in this task.

13.

Write a function/method called **weirdCombination()** that takes two parameters where one is an array and the other one is a head of a linked list. The length of the array and linked list will always be the same. Your function/method should sum the **n-ith** element of the linkedlist with the **ith** element of the linkedlist and sum up all the subtractions to return at the end. Check the sample output below for more clarification:

Sample Given Array & LinkedList	Sample Returned Value				
linkedList = 10 -> 23 -> 30 -> 14 array = <table><tr><td>15</td><td>10</td><td>56</td><td>65</td></tr></table>	15	10	56	65	-69
15	10	56	65		
	Explanation: It's basically summation of the subtractions. $(10-65)+(23-56)+(30-10)+(14-15) = -69$. Here, the first elements are of the linked lists and the second elements are of the arrays.				

NOTE: No need to write the Node class. Just assume Node class is there with two instance variables; one is elem and the other one is next. **NEGATIVE INDEX** not allowed for python.

14.

Write a function/method called `weirdCombination()` that takes two parameters where one is an array and the other one is a head of a linked list. The length of the array and linked list will always be the same. Your function should subtract the i^{th} element of the linkedlist from the $n-i^{\text{th}}$ element of the array from with the and sum up all the subtractions to return at the end. Check the sample output below for more clarification:

Sample Given Array & LinkedList	Sample Returned Value				
<p>array =</p> <table><tr><td>15</td><td>10</td><td>56</td><td>65</td></tr></table> <p>linkedlist = 10 -> 23 -> 30 -> 14</p>	15	10	56	65	<p>69</p> <p>Explanation: It's basically summation of the subtractions. (65-10)+(56-23)+(10-30)+(15-14) = 69. Here, the first elements are of arrays and the second elements are of the LinkedList.</p>
15	10	56	65		

NOTE: No need to write the Node class. Just assume Node class is there with two instance variables; one is elem and the other one is next. **NEGATIVE INDEX** not allowed for python.

15.

Question: You are given an array where each index contains an integer value. Your task is to find the highest number of times the same value appears consecutively in the array. Consecutive occurrences mean the same number appears in **adjacent indices** one after another.

Write a function `getMaxConsecutiveOccurrence(array)` that takes in an array and returns an integer representing the maximum count of consecutive elements that appear in the list.

Sample Input	Sample Output							
<div>arr =</div> <table><tr><td>1</td><td>1</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td></tr></table>	1	1	4	4	4	2	2	3
1	1	4	4	4	2	2		

Explanation: The output is 3 because the maximum number of consecutive occurrences is 3 (for the element 4).

Sample Input	Sample Output				
<div>arr =<table><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr></table></div>	1	2	3	4	1
1	2	3	4		

Explanation: All elements are unique, so the maximum consecutive occurrence is 1.

16.

Question: You are given a singly linked list where each node contains an integer value. Your task is to find the highest number of times the same value appears consecutively in the list. Consecutive occurrences mean the same number appears in **adjacent nodes** one after another.

Write a function `getMaxConsecutiveOccurrence(LinkedList)` that takes in a linked list and returns an integer representing the maximum count of consecutive elements that appear in the list.

Sample Input	Sample Output
LL = 1 -> 1 -> 4 -> 4 -> 4 -> 2 -> 2	3

Explanation: The output is 3 because the maximum number of consecutive occurrences is 3 (for the element 4).

Sample Input	Sample Output
LL = 1 -> 2 -> 3 -> 4	1

Explanation: All elements are unique, so the maximum consecutive occurrence is 1.

NOTE: You can assume a `nodeAt()` function/method is already implemented; this method/function has one parameter which is the **index** and **returns the reference of the node at that index**. You can call this method/function while solving this task.

17.

You are working on a data processing system that filters sensor readings collected at different times. The readings are stored in a singly linked list, where each node represents a recorded value. Your task is to implement the function `validate_readings(head, low, high)`, which checks if all sensor readings fall within the specified acceptable range [low, high].

- If all values in the linked list are within the range [low, high], print the entire linked list.
- If any value falls outside the range, return "Reading out of range detected."

No other data structures can be used other than linked lists.

Consider that node class is already provided.

Sample Input	Sample Output & Explanation
15 -> 18 -> 10 -> 20 -> 12 low = 10 high = 20	Sample Output: 15 -> 18 -> 10 -> 20 -> 12 Explanation: All values fall within the range [10, 20], so the entire linked list is printed.
25 -> 15 -> 30 -> 18 -> 10 -> 22 low = 10 high = 20	Sample Output: Reading out of range detected. Explanation: The numbers 25, 30, and 22 are outside the range [10, 20], so the function returns "Reading out of range detected."

18.

You are working on a data processing system that filters sensor readings collected at different times. The readings are stored in a singly linked list, where each node represents a recorded value. Your task is to implement the function `validate_readings(head, low, high)`, which checks if all sensor readings fall outside the specified acceptable range `[low, high]`.

- If all values in the linked list are outside the range `[low, high]`, print the entire linked list.
- If any value falls within the range, print "Reading inside the range detected."

No other data structures can be used other than linked lists.

Consider that node class is already provided.

Sample Input	Sample Output & Explanation
25 -> 18 -> 55 -> 20 -> 12 low = 10 high = 20	Sample Output: Reading inside the range detected. Explanation: The numbers 18, 20, and 12 are within the range <code>[10, 20]</code> , so the function prints "Reading inside the range detected."
14 -> 12 -> 17 -> 20 low = 30 high = 40	Sample Output: 14 -> 12 -> 17 -> 20 Explanation: All values fall outside the range <code>[30, 40]</code> , so the entire linked list is printed.

19.

Write a function named `rearrangeNodes` which takes as input the head of a linked list and a value `x` such that all nodes with values less than `x` come before nodes with values greater than or equal to `x`. The nodes should be in their relative order. So, for nodes with values less than `x`, the node that is in an earlier position in the list should come before those that came after it and the same should be followed for those not less than `x`. The function should return the head of the modified linked list.

Consider that the Node class is already provided with the `elem` being the node's value and `next` being the location of the next node.

You are not allowed to create another linked list with new nodes. You have to modify the currently provided list.

Sample Input	Sample Output
head = 1 -> 4 -> 3 -> 2 -> 5 -> 2 x = 3	Sample output: 1 -> 2 -> 2 -> 4 -> 3 -> 5 Explanation: The numbers 1, 2 and 2 are less than the value of 3 while 4, 3 and 5 so the nodes with values 1, 2 and 2 come before the rest. For values less than 3, 1 comes before 2 in the unmodified list so after modification it comes before 3
head = 2 -> 1 x = 2	Sample output: 1 -> 2 Explanation: The number 1 is less than 2 while 2 (in the linked list) is not.

20.

Write a function named **rearrangeNodes** which takes as input the head of a linked list and a value **x** such that all nodes with values greater than **x** come before nodes with values less than or equal to **x**. The nodes should be in their relative order. So, for nodes with values greater than **x**, the node that is in an earlier position in the list should come before those that came after it and the same should follow for nodes with values not greater than **x**. The function should return the head of the modified linked list.

Consider that the **Node** class is already provided with the **elem** being the node's value and **next** being the location of the next node.

You are not allowed to create another linked list with new nodes . You have to modify the currently provided list.

Sample Input	Sample Output
head = 1 -> 4 -> 3 -> 2 -> 5 -> 2 x = 3	Sample output: 4 -> 5 -> 1 -> 3 -> 2 -> 2 Explanation: The numbers 4 and 5 are greater than the value of 3 while 1, 2, 3 and 3 are not. So, the nodes with values 4 and 5 come before the other nodes. For values greater than 3, 4 comes before 5 in the unmodified list so after modification it comes before 5.
head = 2 -> 3 x = 2	Sample output: 3 -> 2 Explanation: The number 3 is greater than 2 while 2 (in the linked list) is not.


21.

Tasks from Book -  Practice Sheet 3 - Linked List.pdf

22.

Tasks from Lab 1 -  Lab 1 : Array and SinglyLinkedList.docx

23.

Tasks from Lab 2 -  Lab 2 - Singly & Doubly LinkedList.docx