Assignment: Computer Science

Computer Science: Problem Solving Assignment - Linked Lists

Due Date: April 5th, 2025 (2025-04-05)

Word Limit: 1000 words (excluding code)

Topic: Linked Lists

This assignment focuses on your understanding and implementation of linked lists in a chosen programming language (e.g., Python, Java, C++, C#). You are required to demonstrate your proficiency in manipulating linked lists through the implementation of various operations and problem-solving exercises. Focus on code clarity, efficiency, and proper commenting. Your submission should include both your code and a concise, well-written report explaining your approach, design choices, and the challenges encountered.

Part 1: Implementation of Basic Linked List Operations (40%)

Implement a singly linked list data structure. Your implementation should include the following functionalities:

1. **Insertion:** Methods for inserting a node at the beginning, at the end, and at a specific position in the list.
2. **Deletion:** Methods for deleting a node at the beginning, at the end, and at a specific position in the list. Handle edge cases (empty list, attempting to delete a non-existent node).
3. **Search:** A method to search for a specific value within the list and return its position (or

indicate if it's not found).

4. **Traversal:** A method to traverse and print the elements of the linked list.

5. **Length:** A method to return the number of nodes in the list.

Part 2: Problem Solving with Linked Lists (60%)

Choose **three** of the following problems to solve using your implemented linked list:

1. **Palindrome Check:** Write a function that checks if a linked list is a palindrome (reads the same forwards and backward).

2. **Merge Two Sorted Lists:** Write a function that takes two sorted linked lists as input and merges them into a single sorted linked list.

3. **Reverse a Linked List:** Write a function that reverses a linked list in-place (without creating a new list).

4. **Detect a Cycle (Loop):** Write a function that detects if a linked list contains a cycle (a node points to a previously visited node). If a cycle exists, return `True`; otherwise, return `False`. Implement using Floyd's Tortoise and Hare algorithm.

5. **Remove Duplicates from a Sorted Linked List:** Write a function that removes duplicate nodes from a sorted linked list, ensuring that only unique values remain.

For each problem you choose, provide:

* A clear description of your algorithm.
* The source code implementing your solution.
* A discussion of the time and space complexity of your algorithm.
* Test cases demonstrating the correctness of your implementation.

AI-Generated Questions (Consider these as supplementary challenges):

1. **Nth Node from the End:** Design an algorithm to find the Nth node from the end of a singly linked list in a single pass (O(n) time complexity).

2. **Intersection of Two Linked Lists:** Given two linked lists, determine if they intersect and if so, find the intersecting node.

3. **Linked List Sorting:** Implement a linked list sorting algorithm (e.g., merge sort, insertion sort) for a linked list containing unsorted integer data. Analyze its time and space complexity.

4. **Circular Linked List Implementation:** Implement a circular linked list (where the last node points back to the first). Include methods for insertion, deletion, and traversal.

5. **Doubly Linked List Implementation:** Implement a doubly linked list (each node contains pointers to both the next and previous nodes). Include methods for insertion, deletion, and traversal. Compare and contrast the performance of doubly linked lists compared to singly linked lists for various operations.

Submission:

Submit a single PDF document containing:

* A title page with your name, student ID, and the assignment title.
* A concise report (max 1000 words) explaining your approach, design choices, challenges, and

time/space complexity analysis for each problem solved. Include screenshots or other visual aids to enhance understanding if needed.

* Your source code (well-commented) for both Part 1 and Part 2. Include clear instructions on how to compile and run your code.

This assignment assesses your understanding of data structures, algorithm design, and problem-solving skills. Ensure your code is well-documented, efficient, and adheres to good programming practices. Late submissions will be penalized.