# data structures – Assessment Bank

Generated on: 2025-06-16 13:50:46

1. Design a linked list implementation that supports both insertion and deletion at the head and tail in O(1) time complexity. Include code snippets to illustrate.  
  
2. Create a function to reverse a singly linked list iteratively. Provide comments explaining the algorithm's steps.  
  
3. Develop a linked list-based implementation of a stack data structure. Show how push and pop operations would be implemented.  
  
4. Design a linked list to efficiently store and retrieve polynomial terms. Explain how addition of two polynomials would be performed.  
  
5. Construct a circular linked list and write a function to detect the presence of a cycle within it using Floyd's Tortoise and Hare algorithm.  
  
6. Create a doubly linked list and implement a function to delete a node given only a pointer to that node.  
  
7. Design a merge function that takes two sorted linked lists as input and returns a single sorted linked list containing all the elements.  
  
8. Develop a function to find the middle node of a singly linked list in a single pass.  
  
9. Create a linked list that supports the insertion of a node at a specified position. Handle edge cases like insertion at the beginning, end, and invalid position.  
  
10. Design a function to remove all duplicates from a sorted linked list.  
  
11. Create a linked list representation of a sparse matrix. Explain how to efficiently perform matrix addition using this representation.  
  
12. Implement a function to detect and remove loops in a singly linked list, if present.  
  
13. Develop a function to copy a linked list while preserving the original list's structure.  
  
14. Create a linked list implementation for a Least Recently Used (LRU) cache. Demonstrate how to handle cache misses and evictions.  
  
15. Design a function to partition a linked list around a given value x, such that all nodes less than x come before all nodes greater than or equal to x.

---  
Watermark: TeachMate AI | Version 1.0