Linked List – Problems [60 marks, 120 minutes]

Note:
- Feel free to modify the function defintion as per your language of choice.
- Simply complete the function defintion, don't write the entire code starting with main.
- This is a pen/paper assignment. Do not write the code in systems.
- Include a 2 liner explanation for your code and state down assumptions clearly, if any.

1. Remove duplicates from a sorted linked list. [5 marks]
   void removeDuplicates(node* head);
   Example:
   Input: 11->11->11->21->43->43->60->NULL
   Output: 11->21->43->60->NULL

2. Check if a singly linked-list is palindrome or not. Expected Space Complexity is O(1) [10 marks]
   bool isPalindrome(node *head);

3. Reverse a doubly linked list and return the pointer to the new head. [10 marks]
   node *reverse(node * head);

4. Segregate even and odd numbers in a Linked List of integers, keeping the order of even and odd numbers same. [10 marks]
   void segregateEvenOdd(node **head);

   Examples:
   Input: 17->15->8->12->10->5->4->1->7->6->NULL
   Output: 8->12->10->4->6->17->15->5->1->7->NULL

   Input: 8->12->10->5->4->1->6->NULL
   Output: 8->12->10->4->6->5->1->NULL

5. Reverse a Linked List in alternate groups of given size and return the pointer to the new head node. [10 marks]
   node* reverseChunks(node *head, int k);

   Example:
   Input: 1->2->3->4->5->6->7->8->9->10->11->12->13->14->NULL and k = 3
   Output: 3->2->1->4->5->6->9->8->7->10->11->12->14->13->NULL.
   Input: 1->2->3->4->5->6->7->8->NULL and k = 5
   Output: 5->4->3->2->1->6->7->8->NULL.

6. Implement LRU cache. [15 marks]
   node * LRU(int page[], int n, int maxCacheSize); Or
   node *LRU(vector<int> page, int maxCacheSize);