Tutorial 4
Linked Lists

Lab 3 Solutions

Q1.) Write a removeDuplicates() function which takes a list and deletes any duplicate nodes from the list. Assume that the list is not sorted.

For example if the linked list is 12->11->12->21->41->43->21 then removeDuplicates() should convert the list to 12->11->21->41->43.

A Simple Approach

1. Use 2 loops
2. Outer loop picks elements of linked list one by one
3. Inner loop compares the element picked by the outer loop with rest of the elements

```
 void remove_duplicates() {
        Node ptr1 = null, ptr2 = null, dup =
null;
        ptr1 = head;

        /* Pick elements one by one */
        while (ptr1 != null && ptr1.next != null)
{
            ptr2 = ptr1;

            /* Compare the picked element
with rest of the elements */
            while (ptr2.next != null) {

                /* If duplicate then delete it */
                if (ptr1.data == ptr2.next.data) {

                    /* sequence of steps is
important here */
                    dup = ptr2.next;
                    ptr2.next = ptr2.next.next;
                    System.gc();
                } else /* This is tricky */ {
                    ptr2 = ptr2.next;
                }
            }
            ptr1 = ptr1.next;
        }
    }
```

https://www.geeksforgeeks.org/remove-duplicates-from-an-unsorted-linked-list/

Q2. Reverse alternate K nodes in a Singly Linked List

Example:

Inputs: 1->2->3->4->5->6->7->8->9->NULL and k = 3

Output: 3->2->1->4->5->6->9->8->7->NULL.

Method

kAltReverse(struct node *head, int k)

1) Reverse first k nodes.
2) In the modified list head points to the kth node. So change next of head to (k+1)th node
3) Move the current pointer to skip next k nodes.
4) Call the kAltReverse() recursively for rest of the n - 2k nodes.
5) Return new head of the list.

https://www.geeksforgeeks.org/reverse-alternate-k-nodes-in-a-singly-linked-list//

```
/* Reverses alternate k nodes and
     returns the pointer to the new head
node */
    Node kAltReverse(Node node, int k) {
        Node current = node;
        Node next = null, prev = null;
        int count = 0;

        /*1) reverse first k nodes of
the linked list */
        while (current != null && count
< k) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
            count++;
        }
 /* 2) Now head points to the kth node.
So change next
         of head to (k+1)th node*/
        if (node != null) {
            node.next = current;
        }
 /* 3) We do not want to reverse next k
nodes. So move the current
         pointer to skip next k nodes */
        count = 0;
        while (count < k - 1 && current
!= null) {
            current = current.next;
            count++;
        }

        /* 4) Recursively call for the
list starting from current->next.
         And make rest of the list as
next of first node */
        if (current != null) {
            current.next =
kAltReverse(current.next, k);
        }

        /* 5) prev is new head of the
input list */
        return prev;
    }
```

https://www.geeksforgeeks.org/reverse-alternate-k-nodes-in-a-singly-linked-list//

Q3.) Write a function to check if a singly linked list is palindrome or not

Example:

A->B->C->B->A

Palindrome

Method

1) Get the middle of the linked list.

2) Reverse the second half of the linked list.

3) Check if the first half and second half are identical.

4) Construct the original linked list by reversing the second half again and attaching it back to the first half

https://www.geeksforgeeks.org/function-to-check-if-a-singly-linked-list-is-palindrome//

Q4.) Append one list at the end of another list

Try yourself [Refer the lecture slides]

Q5.) Merge 2 sorted lists to get a single sorted list

Q5.) Merge 2 sorted lists to get a single sorted list

```
Node merge( Node p, Node q ) {
        if ( p == null)
           return q;
        else if ( q == null)
           return p;
        else if (p.item < q.item) {
              p.next = merge( p.next, q );
              return p;
           }
           else {
              q.next = merge( p, q.next );
              return q;
           }
        }
```

Homework Question

Pairwise swap elements of a given linked list

Input: 1->2->3->4->5

Output: 2->1->4->3->5,

Input: 1->2->3->4->5->6

Output: 2->1->4->3->6->5