Segregate even and odd nodes in a Linked List

Given a Linked List of integers, write a function to modify the linked list such that all even numbers appear before all the odd numbers in the modified linked list. Also, keep the order of even and odd numbers same.

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

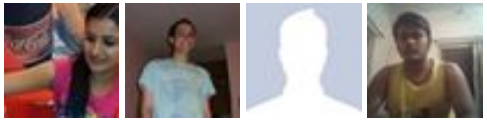
// If all numbers are even then do not change the list
Input: 8->12->10->NULL
Output: 8->12->10->NULL

// If all numbers are odd then do not change the list
Input: 1->3->5->7->NULL
Output: 1->3->5->7->NULL

Method 1
The idea is to get pointer to the last node of list. And then traverse the list starting from the head node and move the odd valued nodes from their current position to end of the list.

Thanks to blunderboy for suggesting this method.

Algorithm:
…1) Get pointer to the last node.



…2) Move all the odd nodes to the end.
……..a) Consider all odd nodes before the first even node and move them to end.
……..b) Change the head pointer to point to the first even node.
……..b) Consider all odd nodes after the first even node and move them to the end.

```
#include <stdio.h>
#include <stdlib.h>

/* a node of the singly linked list */
struct node
{
    int data;
    struct node *next;
};
```

```
void segregateEvenOdd(struct node **head_ref)
{
    struct node *end = *head_ref;

    struct node *prev = NULL;
    struct node *curr = *head_ref;

    /* Get pointer to the last node */
    while (end->next != NULL)
        end = end->next;

    struct node *new_end = end;

    /* Consider all odd nodes before the first even node
       and move then after end */
    while (curr->data %2 != 0 && curr != end)
    {
        new_end->next = curr;
        curr = curr->next;
        new_end->next->next = NULL;
        new_end = new_end->next;
    }

    // 10->8->17->17->15
    /* Do following steps only if there is any even node */
    if (curr->data%2 == 0)
    {
        /* Change the head pointer to point to first even node */
        *head_ref = curr;

        /* now current points to the first even node */
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
        while (curr != end)
        {
            if ( (curr->data)%2 == 0 )
            {
                prev = curr;
                curr = curr->next;
            }
            else
            {
                /* break the link between prev and current */
                prev->next = curr->next;

                /* Make next of curr as NULL  */
                curr->next = NULL;

                /* Move curr to end */
                new_end->next = curr;

                /* make curr as new end of list */
                new_end = curr;

                /* Update current pointer to next of the moved node */
                curr = prev->next;
            }
        }
    }

    /* We must have prev set before executing lines following this
       statement */
    else prev = curr;

    /* If there are more than 1 odd nodes and end of original list is
      odd then move this node to end to maintain same order of odd
      numbers in modified list */
    if (new_end!=end && (end->data)%2 != 0)
    {
        prev->next = end->next;
        end->next = NULL;
        new_end->next = end;
    }
    return;
}

/* UTILITY FUNCTIONS */
/* Function to insert a node at the beginning of the Doubly Linked Lis
void push(struct node** head_ref, int new_data)
{
```

```
    /* allocate node */
    struct node* new_node =
        (struct node*) malloc(sizeof(struct node));

    /* put in the data  */
    new_node->data  = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref)    = new_node;
}

/* Function to print nodes in a given doubly linked list
   This function is same as printList() of singly linked lsit */
void printList(struct node *node)
{
    while (node!=NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Drier program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;

    /* Let us create a sample linked list as following
      0->2->4->6->8->10->11 */

    push(&head, 11);
    push(&head, 10);
    push(&head, 8);
    push(&head, 6);
    push(&head, 4);
    push(&head, 2);
    push(&head, 0);

    printf("\n Original Linked list ");
    printList(head);

    segregateEvenOdd(&head);
```


Recent Comments

```
    printf("\n Modified Linked list ");
    printList(head);

    return 0;
}
```

Output:

```
Original Linked list 0 2 4 6 8 10 11
Modified Linked list 0 2 4 6 8 10 11
```

Time complexity: O(n)

Method 2

The idea is to split the linked list into two: one containing all even nodes and other containing all odd nodes. And finally attach the odd node linked list after the even node linked list.

To split the Linked List, traverse the original Linked List and move all odd nodes to a separate Linked List of all odd nodes. At the end of loop, the original list will have all the even nodes and the odd node list will have all the odd nodes. To keep the ordering of all nodes same, we must insert all the odd nodes at the end of the odd node list. And to do that in constant time, we must keep track of last pointer in the odd node list.

Time complexity: O(n)

Please write comments if you find the above code/algorithm incorrect, or find other ways to solve the same problem.

Related Tpoics:

- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links
- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- Swap Kth node from beginning with Kth node from end in a Linked List

Writing code in comment? Please use **ideone.com** and share the link here.