EE 2331 Data Structures and Algorithms, Semester B, 2008/09

Tutorial 7: – Linked Lists Reversion

Week 7 (4th March, 2010)

The tasks of tutorial exercises are divided into three levels. Level 1 is the basic tasks. You should have enough knowledge to complete them after attending the lecture. Level 2 is the advanced tasks. You should able to tackle them after revision. Level 3 is the challenge tasks which may be out of the syllabus and is optional to answer. <u>*I expect you to complete task A in the tutorial.*</u>

Outcomes of this assignment

1. Able to reverse linked lists by recursion
2. Able to generalize a problem using recursion

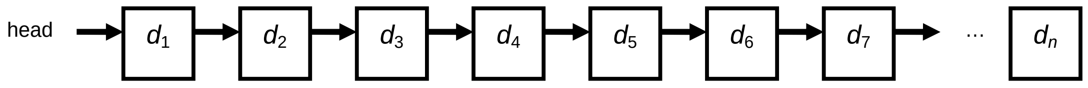
Reversing a linked list is a common operation in computer programming. There are several ways to reverse a linked list. The typical method is to manipulate the linked list pointers directly. Other methods like using recursion or with the help of a stack are also common.

In this assignment, you are asked to write a C program to reverse a singly linear linked list using recursion.

The node structure of the linked list is defined as follows:

```
typedef struct _node{
   int data;                  //the value of the node
   struct _node *next;        //point to next node
} Node;
```

The list can be visualized as below:



d_i: the i th node

Figure 1.The representation of linear singly linked list without dummy header node.

Task A (Level 2): Reverse by recursion

Complete the C function **Node * reverse_by_recur(Node *head)** that accepts the head of a linked list and reverses the list by recursion. The function should return the pointer of the tail node. Please be noted that you are expected to reverse the list by changing the pointers, but not swapping the integer values between nodes.

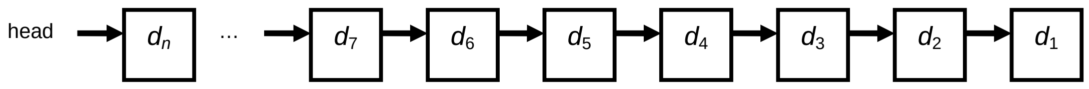


Figure 2.The linked list after reversion.

Expected Output:

```
Enter the number of nodes: 8
Enter your action ( 1) task a, 2) task b): 1

The original inked list:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> [7] -> [8] -> NULL

The reversed linked list:
[8] -> [7] -> [6] -> [5] -> [4] -> [3] -> [2] -> [1] -> NULL

Reverse again:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> [7] -> [8] -> NULL
```

```
Enter the number of nodes: 0
Enter your action ( 1) task a, 2) task b): 1

The original inked list:
NULL

The reversed linked list:
NULL

Reverse again:
NULL
```

```
Enter the number of nodes: 1
Enter your action ( 1) task a, 2) task b): 1

The original inked list:
[1] -> NULL

The reversed linked list:
[1] -> NULL

Reverse again:
[1] -> NULL
```

Discussion: What are the respectively time complexity and space complexity of the algorithm?

Task B (Level 2): Reverse two nodes as a group by recursion

Complete the C function **Node * reverse_group_by_recur(Node *head)** that accepts the head of a linked list and reverses the nodes as the following by using recursion.

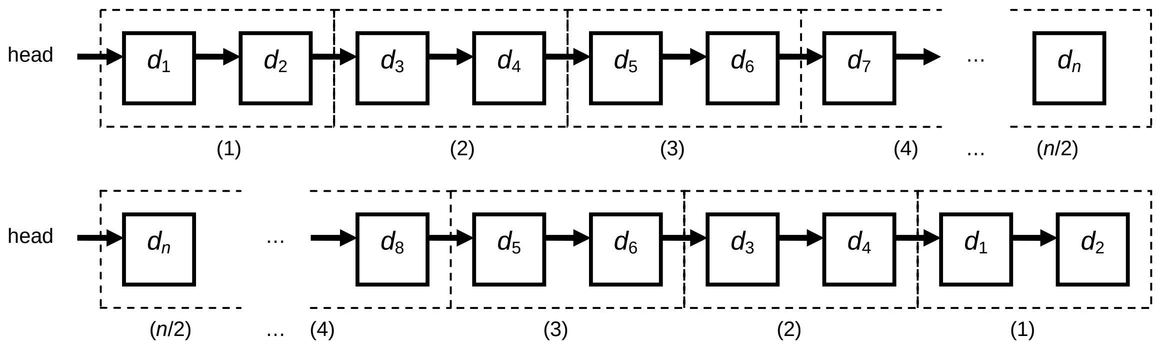


Figure 3.The linked list after reversion.

Expected Output: **<u>Even number of nodes</u>**

```
Enter the number of nodes: 8
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> [7] -> [8] -> NULL

The reversed linked list:
[7] -> [8] -> [5] -> [6] -> [3] -> [4] -> [1] -> [2] -> NULL

Reverse again:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> [7] -> [8] -> NULL
```

```
Enter the number of nodes: 6
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> NULL

The reversed linked list:
[5] -> [6] -> [3] -> [4] -> [1] -> [2] -> NULL

Reverse again:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> NULL
```

<u>Odd number of nodes</u>

```
Enter the number of nodes: 7
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
[1] -> [2] -> [3] -> [4] -> [5] -> [6] -> [7] -> NULL

The reversed linked list:
[7] -> [5] -> [6] -> [3] -> [4] -> [1] -> [2] -> NULL

Reverse again:
[2] -> [4] -> [1] -> [6] -> [3] -> [7] -> [5] -> NULL
```

```
Enter the number of nodes: 5
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
[1] -> [2] -> [3] -> [4] -> [5] -> NULL

The reversed linked list:
[5] -> [3] -> [4] -> [1] -> [2] -> NULL

Reverse again:
[2] -> [4] -> [1] -> [5] -> [3] -> NULL
```

Special number of nodes

```
Enter the number of nodes: 0
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
NULL

The reversed linked list:
NULL

Reverse again:
NULL
```

```
Enter the number of nodes: 1
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
[1] -> NULL

The reversed linked list:
[1] -> NULL

Reverse again:
[1] -> NULL
```

```
Enter the number of nodes: 2
Enter your action ( 1) task a, 2) task b): 2

The original inked list:
[1] -> [2] -> NULL

The reversed linked list:
[1] -> [2] -> NULL

Reverse again:
[1] -> [2] -> NULL
```