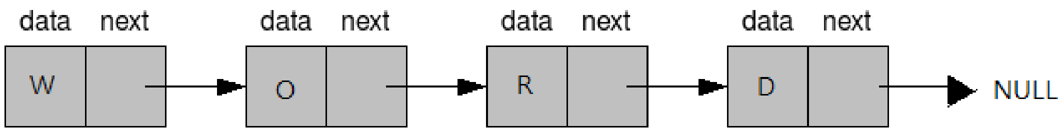
Homework #7

Due date: 18:00, December 5th, Monday, 2016

Problem 1 (80%)

In this assignment, we would like to introduce a data structure called "linked list" to implement string reversing. The structure of linked list is as the following picture. (representing the string "word")



Linked list is constructed by a set of "nodes", each node contains both data(a char in this case) and a pointer pointing to the next node, and together the nodes form the whole string.

You are required to implement the linked list using struct, and in the struct there are a char and a pointer. Also, there are two functions needed, which are "add" and "reverse." Function "add" adds a character to the end of the list, and "reverse" find the reversal of the string in the list. You may change the return type of the function or add some input arguments, but the red part are needed.

```
struct Node
{
    char c;                    // stores a charater of the
    input string
    struct Node *next;          // points to the next node
};

void append(struct Node*, …); // append a character

char* reverse(struct Node*, …);       // reverse the string
```

Your job is to implement a linked list to store the input string, and try to find the reversal of the string. For example, if the input is "word", then you should output "drow".

Requirements

1. Implement linked list using struct and pointer
2. Implement function “add” and “reverse”.
3. Store the input string in the linked list
4. There will be no blank space in the input string
5. You may assume the input to be correct.
6. Plagiarism is not allowed!

Sample run

```
Enter a string: thisisastring
string reversal : gnirtsasisiht

enter a string: ilovecs
string reversal : scevoli

enter a string: bdalab
string reversal : baladb

enter a string: 32418314
string reversal : 41381423

enter a string: ^Z
```

Problem 2 (20%)

Given an integer of sequence, store it as linked list
and sort it descendingly by Bubble Sort.

(Bubble Sort)

Bubble Sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compare each pair of adjacent items and swap them if they are in the wrong order.

The pass through the list is repeated until no swaps are needed or repeated number equals to the number of sequence -1 , which indicates that the list is sorted.

For example, there is a sequence to be sorted, [1, 5, 3 ,4].

First Pass:

[**1, 5**, 3, 4] -> [**5, 1**, 3, 4]

[5, **1, 3**, 4] -> [5, **3, 1**, 4]

[5, 3, **1, 4**] -> [5, 3, **4, 1**]

Second Pass:

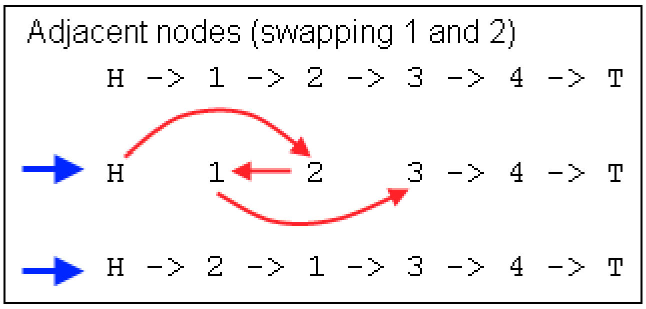
[**5, 3**, 4, 1] -> [**5, 3**, 4, 1]

[5, **3, 4**, 1] -> [5, **4, 3**, 1]

Third Pass:

No swaps

You are required to implement a function called "swap" to swap two adjacent nodes in the linked list. Same as the above, you may change the return type of the function or add some input arguments, but the red part are needed.



Adjacent nodes (swapping 1 and 2)
H -> 1 -> 2 -> 3 -> 4 -> T
H 1 2 3 -> 4 -> T
H -> 2 -> 1 -> 3 -> 4 -> T

```
void swap(struct Node*, …);
```

Requirements

1. Scan a sequence of integer until the input isn't a number
2. Implement function "swap"

Submission

Be sure to upload your source code to E3 by the due date and name your file as "**xxxxxxx_hw7.c**", where **xxxxxxx** is your student ID.

Sample run

```
Enter a sequence of integer:
15
9
2
2
6
1
19
13
F
Output: 19 15 13 9 6 2 2 1
```