Problem Statement:-

Given a linked list, swap every two adjacent nodes and return its head.

For example,  
Given 1->2->3->4, you should return the list as 2->1->4->3.

Your algorithm should use only constant space. You may not modify the values in the list, only nodes itself can be changed.

Code:

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     struct ListNode \*next;

 \* };

 \*

 \* typedef struct ListNode listnode;

 \*

 \* listnode\* listnode\_new(int val) {

 \*     listnode\* node = (listnode \*) malloc(sizeof(listnode));

 \*     node->val = val;

 \*     node->next = NULL;

 \*     return node;

 \* }

 \*/

/\*\*

 \* @input A : Head pointer of linked list

 \*

 \* @Output head pointer of list.

 \*/

listnode\* swapPairs(listnode\* A) {

    listnode\* temp=A;

    while(temp!= NULL && temp->next!= NULL){

        swap(&temp->val,&temp->next->val);

        temp=temp->next->next;

    }

    return A;

}

void swap(int\* a,int\* b){

     int temp;

     temp=\*a;

     \*a=\*b;

     \*b=temp;

 }

Result:-



