

## Pairwise swap elements of a given linked list by changing links

Given a singly linked list, write a function to swap elements pairwise. For example, if the linked list is 1->2->3->4->5->6->7 then the function should change it to 2->1->4->3->6->5->7, and if the linked list is 1->2->3->4->5->6 then the function should change it to 2->1->4->3->6->5

This problem has been discussed [here](#). The solution provided there swaps data of nodes. If data contains many fields, there will be many swap operations. So changing links is a better idea in general. Following is a C implementation that changes links instead of swapping data.

## C

```
/* This program swaps the nodes of linked list rather than swapping the
field from the nodes.
Imagine a case where a node contains many fields, there will be plenty
of unnecessary swap calls. */

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

/* A linked list node */
struct node
{
    int data;
    struct node *next;
};

/* Function to pairwise swap elements of a linked list */
void pairWiseSwap(struct node **head)
{
    // If linked list is empty or there is only one node in list
    if (*head == NULL || (*head)->next == NULL)
        return;

    // Initialize previous and current pointers
    struct node *prev = *head;
    struct node *curr = (*head)->next;

    *head = curr; // Change head before proceeding

    // Traverse the list
    while (true)
    {
        struct node *next = curr->next;
        curr->next = prev; // Change next of current as previous node

        // If next NULL or next is the last node
        if (next == NULL || next->next == NULL)
        {
            prev->next = next;
            break;
        }

        // Change next of previous to next next
        prev->next = next->next;

        // Update previous and curr
        prev = next;
        curr = prev->next;
    }
}

/* Function to add a node at the beginning of Linked List */
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 linked list */
void printList(struct node *node)
{
    while(node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Driver program to test above function */
int main()
{
    struct node *start = NULL;

    /* The constructed linked list is:
    1->2->3->4->5->6->7 */
    push(&start, 7);
    push(&start, 6);
    push(&start, 5);
    push(&start, 4);
    push(&start, 3);
    push(&start, 2);
    push(&start, 1);

    printf("\n Linked list before calling pairWiseSwap() ");
    printList(start);

    pairWiseSwap(&start);

    printf("\n Linked list after calling pairWiseSwap() ");
    printList(start);

    getchar();
    return 0;
}

```

## Java

```

// Java program to swap elements of linked list by changing links

class LinkedList {

    static Node head;

    static class Node {

        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }
}

```

```

}

/* Function to pairwise swap elements of a linked list */
Node pairWiseSwap(Node node) {

    // If linked list is empty or there is only one node in list
    if (node == null || node.next == null) {
        return null;
    }

    // Initialize previous and current pointers
    Node prev = node;
    Node curr = node.next;

    node = curr; // Change head before proceeding

    // Traverse the list
    while (true) {
        Node next = curr.next;
        curr.next = prev; // Change next of current as previous node

        // If next NULL or next is the last node
        if (next == null || next.next == null) {
            prev.next = next;
            break;
        }

        // Change next of previous to next next
        prev.next = next.next;

        // Update previous and curr
        prev = next;
        curr = prev.next;
    }
    return node;
}

/* Function to print nodes in a given linked list */
void printList(Node node) {
    while (node != null) {
        System.out.print(node.data + " ");
        node = node.next;
    }
}

// Driver program to test above functions
public static void main(String[] args) {

    /* The constructed linked list is:
    1->2->3->4->5->6->7 */
    LinkedList list = new LinkedList();
    list.head = new Node(1);
    list.head.next = new Node(2);
    list.head.next.next = new Node(3);
    list.head.next.next.next = new Node(4);
    list.head.next.next.next.next = new Node(5);
    list.head.next.next.next.next.next = new Node(6);
    list.head.next.next.next.next.next.next = new Node(7);

    System.out.println("Linked list before calling pairwiseSwap() ");
    list.printList(head);
    Node st = list.pairWiseSwap(head);
    System.out.println("");
    System.out.println("Linked list after calling pairwiseSwap() ");
    list.printList(st);
    System.out.println("");
}
}

```

// This code has been contributed by Mayank Jaiswal

Output:

```
Linked list before calling pairWiseSwap() 1 2 3 4 5 6 7
Linked list after calling pairWiseSwap() 2 1 4 3 6 5 7
```

Time Complexity: Time complexity of the above program is  $O(n)$  where  $n$  is the number of nodes in a given linked list. The while loop does a traversal of the given linked list.

Following is **recursive implementation** of the same approach. We change first two nodes and recur for the remaining list. Thanks to geek and omer salem for suggesting this method.

## C

```
/* This program swaps the nodes of linked list rather than swapping the
field from the nodes.
Imagine a case where a node contains many fields, there will be plenty
of unnecessary swap calls. */

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

/* A linked list node */
struct node
{
    int data;
    struct node *next;
};

/* Function to pairwise swap elements of a linked list.
It returns head of the modified list, so return value
of this node must be assigned */
struct node *pairWiseSwap(struct node* head)
{
    // Base Case: The list is empty or has only one node
    if (head == NULL || head->next == NULL)
        return head;

    // Store head of list after two nodes
    struct node* remaing = head->next->next;

    // Change head
    struct node* newhead = head->next;

    // Change next of second node
    head->next->next = head;

    // Recur for remaining list and change next of head
    head->next = pairWiseSwap(remaing);

    // Return new head of modified list
    return newhead;
}

/* Function to add a node at the beginning of Linked List */
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 linked list */
void printList(struct node *node)
{
    while(node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Druver program to test above function */
int main()
{
    struct node *start = NULL;

    /* The constructed linked list is:
    1->2->3->4->5->6->7 */
    push(&start, 7);
    push(&start, 6);
    push(&start, 5);
    push(&start, 4);
    push(&start, 3);
    push(&start, 2);
    push(&start, 1);

    printf("\n Linked list before calling pairWiseSwap() ");
    printList(start);

    start = pairWiseSwap(start); // NOTE THIS CHANGE

    printf("\n Linked list after calling pairWiseSwap() ");
    printList(start);

    return 0;
}

```

## Java

```

// Java program to swap elements of linked list by changing links

class LinkedList {

    static Node head;

    static class Node {

        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }

    /* Function to pairwise swap elements of a linked list.
    It returns head of the modified list, so return value
    of this node must be assigned */
    Node pairWiseSwap(Node node) {

        // Base Case: The list is empty or has only one node
        if (node == null || node.next == null) {
            return node;
        }

        // Store head of list after two nodes
        Node remaing = node.next.next;

        // Change head

```

```

Node newhead = node.next;

// Change next of second node
node.next.next = node;

// Recur for remaining list and change next of head
node.next = pairWiseSwap(remaing);

// Return new head of modified list
return newhead;
}

/* Function to print nodes in a given linked list */
void printList(Node node) {
    while (node != null) {
        System.out.print(node.data + " ");
        node = node.next;
    }
}

// Driver program to test above functions
public static void main(String[] args) {

    /* The constructed linked list is:
    1->2->3->4->5->6->7 */
    LinkedList list = new LinkedList();
    list.head = new Node(1);
    list.head.next = new Node(2);
    list.head.next.next = new Node(3);
    list.head.next.next.next = new Node(4);
    list.head.next.next.next.next = new Node(5);
    list.head.next.next.next.next.next = new Node(6);
    list.head.next.next.next.next.next.next = new Node(7);

    System.out.println("Linked list before calling pairWiseSwap() ");
    list.printList(head);
    head = list.pairWiseSwap(head);
    System.out.println("");
    System.out.println("Linked list after calling pairWiseSwap() ");
    list.printList(head);
    System.out.println("");

}
}

```

```

Linked list before calling pairWiseSwap() 1 2 3 4 5 6 7
Linked list after calling pairWiseSwap() 2 1 4 3 6 5 7

```