Write a recursive function to print reverse of a Linked List

Given a linked list, print reverse of it using a recursive function. For example, if the given linked list is 1->2->3->4, then output should be 4->3->2->1.

Note that the question is only about printing the reverse. To reverse the list itself see this

Difficulty Level: Rookie

Algorithm

```
printReverse(head)
  1. call print reverse for hed->next
  2. print head->data
```

Implementation:

C

```c
// C program to print reverse of a linked list
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* Function to reverse the linked list */
void printReverse(struct node* head)
{
    // Base case
    if (head == NULL)
       return;

    // print the list after head node
    printReverse(head->next);

    // After everything else is printed, print head
    printf("%d  ", head->data);
}

/*UTILITY FUNCTIONS*/
/* Push a node to linked list. Note that this function
  changes the head */
void push(struct node** head_ref, char 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 pochar to the new node */
    (*head_ref)    = new_node;
}

/* Drier program to test above function*/
int main()
{
    // Let us create linked list 1->2->3->4
    struct node* head = NULL;
    push(&head, 4);
    push(&head, 3);
    push(&head, 2);
    push(&head, 1);

    printReverse(head);
    return 0;
}
```

Java

```
// Java program to print reverse of a linked list
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }

    /* Function to print reverse of linked list */
    void printReverse(Node head)
    {
        if (head == null) return;

        // print list of head node
        printReverse(head.next);

        // After everything else is printed
        System.out.print(head.data+" ");
    }

    /* Utility Functions */

    /* Inserts a new Node at front of the list. */
    public void push(int new_data)
    {
        /* 1 & 2: Allocate the Node &
                  Put in the data*/
        Node new_node = new Node(new_data);

        /* 3. Make next of new Node as head */
        new_node.next = head;

        /* 4. Move the head to point to new Node */
        head = new_node;
    }

    /*Drier function to test the above methods*/
    public static void main(String args[])
    {
        // Let us create linked list 1->2->3->4
        LinkedList llist = new LinkedList();
        llist.push(4);
        llist.push(3);
        llist.push(2);
        llist.push(1);

        llist.printReverse(llist.head);
    }
}
/* This code is contributed by Rajat Mishra */
```

Output:

```
4 3 2 1
```

Time Complexity: O(n)