A linked list is given. The goal is to reverse the list, but unfortunately there are elements that shouldn't be moved: every k-th element should be kept where it is, and the rest of the nodes should be reversed. Return the changed linked list.

While I can't enforce this, try to not create new nodes but only reuse / move existing nodes.

**Example**  
For list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and k = 2, the output should be  
reverseLinkedListExceptEveryK(list) = [9, 2, 7, 4, 5, 6, 3, 8, 1, 10].  
As 2, 4, 6, 8, 10 are fixed and 1, 3, 5, 7, 9 are reversed into 9, 7, 5, 3, 1 in place.

* **[execution time limit] 3 seconds (cs)**
* **[input] linkedlist.integer list**

A linked list of elements. Elements may repeat.

*Guaranteed constraints:*  
list.length < 1000,  
0 ≤ list[i] ≤ 1000.

* **[input] integer k**

The interval of elements to keep in place.

*Guaranteed constraints:*  
0 < k ≤ list.length.

* **[output] linkedlist.integer**

The reversed linked list, except every k index which should correspond to the original list.

Remember that the tail should point to undefined and therefore needs to be unset!~

**[C#] Syntax Tips**

// Prints help message to the console

// Returns a string

**string** **helloWorld**(**string** name) {

Console.Write("This prints to the console when you Run Tests");

**return** "Hello, " + name;

}

<https://codefights.com/challenge/nQLwdrgCmxmDw7PCC>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp40

{

class Program

{

// Definition for singly-linked list:

class ListNode<T>

{

public T value { get; set; }

public ListNode<T> next { get; set; }

}

//static ListNode<int> reverseLinkedListExceptEveryK(ListNode<int> list, int k)

//{

// List<int> variable = new List<int>();

// List<int> fijo = new List<int>();

// int cont = 1;

// while (list.next != null)

// {

// //a.Add(list.value);

// if (cont == k)

// {

// cont = 1;

// fijo.Add(list.value);

// }

// else

// {

// variable.Add(list.value);

// cont++;

// }

// list = list.next;

// }

// variable.Reverse();

// List<int> ans = new List<int>();

// int cfijo = 0, cvar = 0;

// for (int i = 0; i < fijo.Count + variable.Count; i++)

// {

// if ((i + 1) % k == 0)

// {

// ans.Add(fijo[cfijo]);

// if (cfijo < fijo.Count)

// {

// cfijo++;

// }

// }

// else

// {

// ans.Add(variable[cvar++]);

// if (cvar < variable.Count)

// {

// cvar++;

// }

// }

// }

// ListNode<int> r = new ListNode<int>();

// for(int i =0; i<ans.Count; i++)

// {

// r.value = ans[i];

// }

// return r;

//}

//solucion alex rafi

int i, j, t;

ListNode<int> reverseLinkedListExceptEveryK(ListNode<int> l, int k)

{

var a = new List<int>();

var p = l;

while (p != null)

{

a.Add(p.value);

p = p.next;

}

j = a.Count - 1;

for (; i < a.Count / 2; i++)

{

if ((i + 1) % k == 0) continue;

while ((j + 1) % k == 0) j--;

t = a[i];

a[i] = a[j];

a[j] = t;

j--;

}

p = l;

foreach (int b in a)

{

p.value = b;

p = p.next;

}

return l;

}

static void Main(string[] args)

{

int[] list = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

ListNode<int> linked = new ListNode<int>();

for(int i =0; i<list.Length; i++)

{

linked.next = null;

linked.value = 0;

}

for(int i =0; i<list.Length; i++)

{

linked.value = list[i];

linked = linked.next;

}

while (linked.next != null)

{

Console.Write(linked.value + " ");

linked = linked.next;

}

Console.ReadLine();

}

}

}