**LL - K REVERSE**

Given a ****head**** to Linked List ****L****, write a function to reverse the list taking ****k**** elements at a time. Assume ****k**** is a factor of the size of List.

You need not have to create a new list. Just reverse the old one using ****head****.

**Input Format:**

The first line contains 2 space separated integers ****N**** and ****K****

The next line contains ****N**** space separated integral elements of the list.

**Constraints:**

0 <= N <= 10^6 0 <= K <= 10^6

**Output Format**

Display the linkedlist after reversing every k elements

**Sample Input**

9 3

9 4 1 7 8 3 2 6 5

**Sample Output**

1 4 9 3 8 7 5 6 2

Program-

#include<iostream>

using namespace std;

class node{

public:

int data;

node \*next;

node(int d)

{

data=d;

next=NULL;

}

};

void insertattail(node \*&head,int data)

{

if(head==NULL)

{

head=new node(data);

return;

}

node \*temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=new node(data);

return;

}

void buildlist(node \*&head,int n)

{ for(int j=0;j<n;j++)

{

int data;

cin>>data;

insertattail(head,data);

}

}

void print(node \*head)

{

while(head !=NULL)

{

cout<<head->data<<" ";

head=head->next;

}

}

node \*reverse(node \*&head,int k)

{

node \*current=head;

node \*next=NULL;

node \*prev=NULL;

int count=0;

while(current!=NULL && count<k)

{

next=current->next;

current->next=prev;

prev=current;

current=next;

count++;

}

if(next!=NULL)

{

head->next=reverse(next,k);

}

return prev;

}

int main() {

int n,k;

cin>>n>>k;

node \*head=NULL;

buildlist(head,n);

head=reverse(head,k);

print(head);

return 0;

}