```
//
//  List.cpp
//  CppPractice
//
//  Created by Stewart Bracken on 12/9/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//


#include <stdio.h> //NULL
#include <iostream>

//
//  Very simple list class for practicing list algorithms, not intended for
//  production.
//
//  Created by Stewart Bracken on 12/4/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#ifndef InterviewPractice_List_h
#define InterviewPractice_List_h

template <class T>
class List {
    template <class U>
    class Node{
        friend class List<T>;
        T data;
        Node<U>* next;
        Node(U _data):data(_data), next(0){
        }
    };
    typedef Node<T> Node_t;
    Node_t* head;
    Node_t* tail;
public:
    List() : head(0),tail(0) {};
    ~List<T>();
    void insert_back(T data);
    void insert_front(T data);
    //Skip k, then reverse k. Repeat until end.
    void reverse_k(T k);
    void print() const;
};
```



```
#endif


template <class T>
List<T>::~List(){
    Node_t* curr = head;
    while(curr != NULL){
        head = curr->next;
        delete curr;
        curr = head;
    }
}

template <class T>
void List<T>::insert_back(T data){
    Node_t* new_node =new Node_t(data);
    if(tail)
        tail->next = new_node;
    tail =  new_node;
    if(head == 0){
            head = tail;
    }
}

template <class T>
void List<T>::insert_front(T data){
        Node_t* tmp = new Node_t(data);
        tmp->next = head;
        if(head==NULL){tail = tmp; tmp->next=NULL;}
        head = tmp;
    }


template <class T>
void List<T>::reverse_k(T k){
    Node_t* curr = head;
    Node_t* tmp;
    while( curr != NULL ){
        //skip up to k nodes
        for(int i=0; i<k && curr != NULL; ++i){
            curr = curr->next;
        }

        //reverse up to k nodes
```

```
            tmp = curr;
            List<T> reverse;
            for(int i=0; i<k && curr != NULL; ++i){
                reverse.insert_front(curr->data);
                curr = curr->next;
            }
            curr = tmp;
            tmp = reverse.head;
            for(int i=0; i<k && tmp != NULL; ++i){
                curr->data = tmp->data;
                curr = curr->next;
                tmp = tmp->next;
            }
        }
    }

    template<class T>
    void List<T>::print() const{
        for( Node_t* itor = head; itor != 0; itor = itor->next){
            std::cout << itor->data << ", ";
        }
    }

    //
    //  ListKTest_unittest.cpp
    //  CppTests
    //
    //  Created by Stewart Bracken on 12/5/13.
    //  Copyright (c) 2013 Stewart Bracken. All rights reserved.
    //

    #include <gtest/gtest.h>

    #include "List.h"
    //include the template implementation
    #include "List.cpp"

    TEST(ListKTest, SkipKReverseK){
        List<int> list;
        for(int i = 0; i < 20; ++i){
            list.insert_back(i+1);
        }
        list.print();
        list.reverse_k(20);
        list.print();
```



```
    }
```