```
//
//  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>

#include "List.h"

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 << ", ";
    }
}
```