Listas

List.h

```
#ifndef LISTSTACKQUEUE_LIST_H
#define LISTSTACKQUEUE_LIST_H

#include <iostream>
#include <cassert>

using namespace std;

template<typename T>
struct Node{
    T data;
    Node<T>* next;
};

template<typename T>
class List {
private:
    Node<T>* begin;
    int count;
    Node<T>* makeNode(const T& value);
public:
    List();
    ~List();
    void insert(int pos, const T& value);
    void erase(int pos);
    T& get(int pos) const;
    void print() const;
    int size() const;
};
#endif //LISTSTACKQUEUE_LIST_H
```

List.cpp

```
#ifndef LISTSTACKQUEUE_LIST_CPP
#define LISTSTACKQUEUE_LIST_CPP

#include "List.h"

template<typename T>
List<T>:: List(): begin(0), count(0){

}
template<typename T>
List<T>:: ~List(){
    Node<T>* del = begin;
    while (begin){
        begin = begin->next;
        delete del;
        del = begin;
```

```
    }
}
template<typename T>
Node<T>* List<T>::makeNode(const T &value) {
    Node<T>* temp = new Node<T>;
    temp->data = value;
    temp->next = 0;
    return temp;
}
template<typename T>
void List<T>::insert(int pos, const T &value) {
    if(pos < 0 || pos>count){
        cout << "Error! The position is out of range." << endl;
        return;
    }
    Node<T>* add = makeNode(value);
    if(pos == 0){
        add->next = begin;
        begin = add;
    }else{
        Node<T>* cur = begin;
        for(int i=0; i<pos-1; i++){
            cur = cur->next;
        }
        add->next = cur->next;
        cur->next = add;
    }
    count++;
}

template<typename T>
void List<T>::erase(int pos) {
    if(pos < 0 || pos>count){
        cout << "Error! The position is out of range." << endl;
        return;
    }
    if(pos == 0){
        Node<T>* del = begin;
        begin = begin->next;
        delete del;
    }else{
        Node<T>* cur = begin;
        for(int i=0; i<pos-1; i++){
            cur = cur->next;
        }
        Node<T>* del = cur->next;
        cur->next = del->next;
        delete del;
    }
    count--;
}
template<typename T>
T& List<T>::get(int pos) const{
    if(pos < 0 || pos>count-1){
        cout << "Error! The position is out of range." << endl;
```

```
        assert(false);
    }
    if(pos == 0){
        return begin->data;
    }else{
        Node<T>* cur = begin;
        for(int i=0; i<pos; i++){
            cur = cur->next;
        }
        return cur->data;
    }
}
template<typename T>
void List<T>::print() const{
    if(count == 0){
        cout << "List is empty." << endl;
        return;
    }
    Node<T>* cur = begin;
    while(cur){
        cout << cur->data << " ";
        cur = cur->next;
    }
}
template<typename T>
int List<T>::size() const {
    return count;
}
#endif
```

main.cpp

```
#include "List.cpp"

using namespace std;
int main() {
    List<string> list;
    list.insert(0, "nestor");
    list.insert(1, "victor");
    list.insert(2, "Maria");
    list.insert(3, "juan");
    list.insert(4, "pedro");
    list.print();
    cout << "\nSize: " << list.size() << endl;
    cout << "Element (2): " << list.get(2) << endl;

    list.erase(2);
    list.erase(3);
    list.print();
    cout << "\nSize: " << list.size() << endl;
    return 0;
}
```