1. Напишете реализация на свързан списък от цели числа, която е максимално удобна за поддръжката на следните видове итератори:
    a. Класически итератор, който обхожда елементите на списъка в ред на техните индекси
    b. Итератор, който обхожда елементите на списъка в сортиран ред
    c. Итератор, който обхожда елементите на списъка по реда на тяхното въвеждане (Тип опашка)
    d. Итератор, който обхожда елементите на списъка в ред, обратен на тяхното въвеждане (Тип стек)
    e. Итератор, който обхожда елементите на списъка в случаен ред
    f. Итератор, който обхожда уникалните елементи по ред на първото им въвеждане
    g. Итератор - призрак, който обхожда и текущите елементи, и тези, които вече са били премахнати от списъка

```
MyCoolList list;
list.add(1);
list.add(2);
list.add(3);
list.add(-1);
list.add(-5);

MyCoolListIterator iterator = list.getSortIterator();
while (iterator.valid()) {
     cout<<iterator.get();
     iterator.next();
}
//Must print out:
//-5 -1 1 2 3

list.remove(2);

MyCoolListIterator iterator = list.getStackIterator();
while (iterator.valid()) {
     cout<<iterator.get();
     iterator.next();
}
//Must print out:
//-5 -1 2 1
```

```
// ConsoleApplication1.cpp : This file contains the 'main'
function. Program execution begins and ends there.
//

#include <iostream>
#include <list>

using namespace std;

class MyCoolListIterator {
public:
	virtual bool valid() = 0;
	virtual int get() = 0;
	virtual void next() = 0;
};

class SimpleMCLIterator: public MyCoolListIterator {
private:
	list<int> collectionData;
	list<int>::iterator collectionDataIterator;
public:
	SimpleMCLIterator(list<int> data) {
		collectionData = data;
		collectionDataIterator = collectionData.begin();
	}

	bool valid() {
		return collectionDataIterator != collectionData.end();
	}

	int get() {
		return *collectionDataIterator;
	}

	void next() {
		collectionDataIterator++;
	}
};

class MyCoolList {
private:
	list<int> data;
public:
	void add(int value) {
		data.push_back(value);
	}
```

```
	void remove(int index) {
		list<int>::iterator it = data.begin();
		advance(it, index);
		data.erase(it);
	}

	MyCoolListIterator* getSimpleIterator() {
		MyCoolListIterator* mclIterator = new
SimpleMCLIterator(this->data);
		return mclIterator;
	}
};


int main()
{
	MyCoolList mcl;
	mcl.add(1);
	mcl.add(2);
	mcl.add(3);

	MyCoolListIterator* it = mcl.getSimpleIterator();
	while (it->valid()) {
		cout << it->get() << endl;
		it->next();
	}

	delete it;

	mcl.remove(1);

	it = mcl.getSimpleIterator();
	while (it->valid()) {
		cout << it->get() << endl;
		it->next();
	}

	delete it;

}
```