1 Design patterns : sets and iterators

The term "design pattern" describe algorithmic structures that appear very often in many different situations. The idea is to propose an implementation that deal with very abstract objects so that it could be re-use in different cases.

1.1 Example : integer sets and iterators

```
1  class IntIterator {
2  public:
3    virtual bool hasNext() = 0;
4    virtual int next() = 0;
5  };
6
7  class IntSet {
8  public:
9    virtual void add(int k) = 0;
10   virtual IntIterator *iterator() const = 0;
11 };
```

1.2 Exercices

1. Write a function that return the size of a `IntSet` ;
2. propose an implementation of `IntSet` (and of an `IntIterator`) with an array ;
3. propose an implementation of `IntSet` (and of an `IntIterator`) with a linked array.

```
1  int size(const IntSet &set) {
2    int s = 0;
3    IntIterator *i = set.iterator();
4    while(i->hasNext()) { i->next(); s++; }
5    delete i;
6    return s;
7  }
```