Exercise: Model a Linked List in Alloy
Author: Martin Monperrus
Version: 15. Oct. 2012
URL: http://www.monperrus.net/martin/exercise-alloy-linkedlist.pdf

Let's consider the following Alloy specification modeling a linked list.

```
sig LinkedList {
  head : lone Node
}
sig Node {
  next : lone Node
}
```

1/ What does the "lone" mean? Why is it necessary?

2/ What does the following predicate mean? How should the predicate be called?

```
pred _______________[l:LinkedList, n:Node] {
  n = l.head or n in l.head.^next
}
```

3/ Write a predicate called noCycle checking that there is no cycle in the list. The predicate takes one argument of type LinkedList.

4/ Consider the following predicate. It represents a valid insertion in the list.

```
pred add[l:LinkedList, l':LinkedList, new:Node] {
  no new.next
  and not l.contains[new] and l'.contains[new]
}
```

Write a piece of Alloy code checking that "if a list has no cycle, a valid insertion results in new list with no cycle as well". Note that even if you don't have an answer to the previous question, you can use the predicate noCycle in this one.

5/ Removing the condition "no new.next" breaks the property. Give an explanation.

6/ Does this model impose a special implementation, i.e. impose where to actually insert the new node in the list? Justify your answer.