Linked Lists

Use the following datatype for implemeting singly-linked lists in C:

```
typedef int value_type;

typedef struct _cons { value_type head;
                       struct _cons * tail ;
                     } nonEmptyList;

typedef nonEmptyList * list; // NULL used as Nil

#define Nil NULL

/*@
  logic list Nil = \null;

  inductive hasSuffix{L} (list xs, list ys) {
    case hasSuffix_refl {L}:
      ∀ list xs; hasSuffix(xs,xs);
    case hasSuffix_next{L}:
      ∀ list xs, ys;
        \valid(xs) ⇒ hasSuffix(xs→tail , ys) ⇒ hasSuffix(xs,ys);
  }

  predicate finite{L}( list xs) = hasSuffix(xs,Nil);
*/

nonEmptyList * cons(value_type x, list xs) { // NULL used as error
  nonEmptyList * result = malloc(sizeof(struct _cons));
  if (result)
    { result→head = x;
      result→tail = xs;
    }
  return result;
}
```

Note:

- *_cons* is a struct name.

  This name starts with un underscore to document that it should not be used by users of this declaration.
- **struct** *_cons* is a type, the same type as:
- **struct** { **value_type** *head*; **struct** *_cons* ∗ *tail*; }.
- The **typedef** ... *nonEmptyList* makes "*nonEmptyList*" another name for that type.
- ***list* is the type of (possibly empty!) lists!**

The code listings above have been produced by including, before `\begin{document}`, the following:

```
\usepackage{listings}
\usepackage{listingsACSL}
\lstset{%
  language=[ACSL]C,
  frame=single,
  identifierstyle=\slshape,
  columns=flexible}
```

I am using no other packages that might be interfering — if you run into trouble, perhaps comment out some `\usepackage{...}` lines you don't need?