5 Programming in C and C++ (djg11)

A data structure for a LISP-like interpreter is specified in OCaml as follows:

```
type lisp_t =
  | Cons of lisp_t * lisp_t   | SAtom of string
  | Nil                       | IAtom of int
  | Ref of lisp_t ref (* a mutable pointer *)
```

(*a*) Without using C++, implement `lisp_t` in the C language, using one (or more) `struct`s or `union`s. If an `enum` is used to distinguish the variant forms, does it need a `Nil` member? [5 marks]

(*b*) The LISP language would store the list `[1,2]` as `Cons(IAtom 1, Cons(IAtom 2, Nil))`. However, in C, three further representations of lists are arrays of values, arrays of references and the intrusive representation (pointer in the datum/record). Briefly distinguish these three further representations, discussing any advantages for memory or runtime efficiency and whether an item can be in more than one list. [6 marks]

(*c*) General-purpose garbage collectors (like Boehm) work for many simple C coding styles. They operate without knowledge of any user-defined `structs` such as `struct lisp_t`, but they may require heap storage to be allocated with their own variant of `malloc`. Are there likely to be problems for your `lisp_t`? Is the presence of any `enum`s, `union`s, or mutable cells a help or hindrance? [5 marks]

(*d*) If your `struct lisp_t` structure were instead implemented in C++, what changes would be desirable? For C++, is distinguishing variants using an `enum` a good approach? [4 marks]