P1 q6

2000 /cp

Foundations of Computer Science (Paper 2)

Describe how ML lists are represented in storage. Your answer should include diagrams illustrating how the representation of [a,b]@[c,d] is derived from those of the lists [a,b] and [c,d], indicating any sharing of memory. How efficient is the evaluation of [a,b]@l if the list l is very long? [4 marks]

What are cyclic lists and how can they be created in ML? [2 marks]

Describe ML's reference types and their applications. In particular, compare mutable data structures with ordinary ML datatypes. [6 marks]

Code an ML function that takes a mutable list and returns true if the list is cyclic, otherwise returning false. Explain why your function is correct. (Hint: in ML, the equality test p=q is permitted on references and is true if p and q refer to the same location in memory.) [8 marks]

Solution notes

List elements are chained together using internal references, which however are not allowed to be changed. The list [a,b] can be visualized as a rectangle containing *a* and a link to another rectangle containing *b*, but with no further link. The list [c,d] is similar. The result of appending the two lists is to create new rectangles for *a* and *b* such that *b* links into the existing list [c,d]. The cost in space and time of evaluating [a,b]@l is proportional to the length of the first argument, here [a,b].

A cyclic list is one that loops back upon itself. It can only be created in ML using reference types to declare mutable lists. Concatenating the list [1,2,3] with itself will create the cycle [1,2,3,1,2,3,...].

Reference types are covered in the notes. The answer should include the type `ref`, the constructor `ref` for creating references, the assignment operator `:=` and the dereferencing operator, `!`. Applications include traditional (procedural) programming and functions with local state. Related to references are traditional arrays, which can be essential for efficiency. References can also be used to implement linked data structures. While ML's datatypes already provide linked data structures, using references gives the programmer full control over the links. One can concatenate two mutable lists without copying the first argument but instead re-linking its end pointer.

The code for the cycle tester is attached. It is very short, requiring only a polymorphic membership test as an auxiliary function. It works by recording a list of the links it follows as at proceeds down the mutable list: if a link has appeared before then it terminates with `true` and otherwise it continues. If it reaches the end of its argument then it is not cyclic.

```
fun isCyclic visited Nil = false
  | isCyclic visited (Cons(x,mlp)) =
    (mlp mem visited) orelse isCyclic (mlp::visited) (!mlp);
```

Just looking for recurrence of the first link field won't enough. Compare links, not elements.