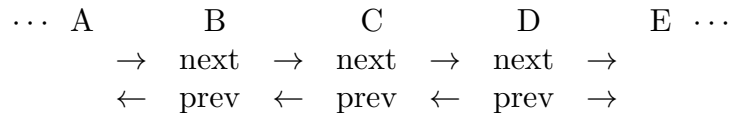## 2010 Paper 3 Question 6
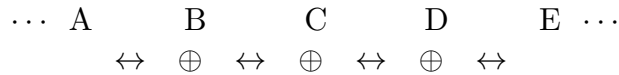
Programming in C and C++

(*a*) Popular programming journal *Obscure C Techniques for Experts* has published a novel way to save space for a doubly-linked list program. Instead of storing two pointers (one next and one previous), this new technique stores a single value: the XOR of *previous* and *next* pointers.

A traditional two-pointer linked list might be illustrated as:

$$\cdots \quad A \qquad B \qquad C \qquad D \qquad E \quad \cdots$$
$$\rightarrow \quad \text{next} \quad \rightarrow \quad \text{next} \quad \rightarrow \quad \text{next} \quad \rightarrow$$
$$\leftarrow \quad \text{prev} \quad \leftarrow \quad \text{prev} \quad \leftarrow \quad \text{prev} \quad \rightarrow$$

In contrast, the proposed new technique stores a bit-wise XOR of the *previous* and *next* pointers within a single field.

$$\cdots \quad A \qquad B \qquad C \qquad D \qquad E \quad \cdots$$
$$\leftrightarrow \quad \oplus \quad \leftrightarrow \quad \oplus \quad \leftrightarrow \quad \oplus \quad \leftrightarrow$$

You have been engaged to provide code examples of this approach for publication.

Ensure your code illustrates the creation and initialization of such a list as well as the insertion, and deletion, of elements from such a list. Additionally, you must provide examples of a forward or backward traversal of the list permitting examination of each element in turn. [15 marks]

(*b*) Comment on this form of linked list. Consider the comparative speed, memory overheads, maintenance and other advantages or disadvantages of the XOR doubly-linked list approach when compared with an approach that stores both previous and next pointers. [5 marks]