```
class List {
    struct Element {
        Element* smaller;
        int key;
        Element* larger;
        Element(int k);
        Element(Element* s, int k, Element* l);
    };
    const int MrBig;
    Element* largest;
    Element* cursor;
    void Kill(void);
public:
    List();
    ~List();
    void Insert(int key);
    void Remove(int key);
    void First(void);
    bool IsEol(void);
    void Next(void);
    int ElementValue(void);
    bool IsMember(int key);

};
```

Figure 1. Example **class** *List*.

1. **class** *List* is implemented using a doubly-linked, circular list with a sentinel. Integers stored in elements are unique.
2. Structure *Element* defines the attributes of an element.
   2.1. Member *smaller* references an element containing a smaller key.
   2.2. Member *key* contains a unique integer value smaller than the maximum integer stored in constant *MrBig*
   2.3. Member *larger* references an element containing a larger key.
3. Constant member *MrBig* contains the largest integer. The value of member *MrBig* is assigned to member *key* in the sentinel element.
4. Member *cursor* points to the current element. Functions *First*, *IsEol*, *Next*, and *Elementvalue* employ member *cursor* to traverse the list without accessing the internal structure of the list.
5. Constructor *List* creates an empty list consisting of the sentinel.
6. Destructor *~List* removes all elements on the list including the sentinel.
7. Member function *Kill* removes all elements on the list except the sentinel.
8. Member function *Insert* creates a new element if parameter *key* is unique. Parameter *key* is assigned to member *key* in the new element. The new element is placed in the list such that elements having smaller keys are found by following the *smaller* link and elements having larger keys are found by following the *larger* link.
9. Member function *Remove* deletes the element having a key that matches parameter *key*. If a match cannot be found no action is taken on the list.
10. Member function *First* positions the *cursor* the smallest element on the list.
11. Member function *IsEol* (*I*s *E*nd *o*f *l*ist) determines if the cursor is positioned on the largest (the sentinel) element on the list.
12. Member function *Next* moves the cursor to the element containing the next larger element on the list.
13. Member function *ElementValue* returns the value of key in the element referenced by the cursor.
14. Member function *member* determines if the list contains an element that has a value that matches parameter *key*.