```
struct ListException {
    ListException(char* m)
    {   cout << endl;
        cout << "I am the List and I am " << m << ".";
        cout << endl;
    }
};
template <class T>
class List {
    int size;           //Number of available elements
    int count;          //Number of occupied elements
                        //Index of the element having the largest value
    int cursor;         //Index of the current element
    T* L;               //Points to storage for the List
    //-----------------------------------------------------------------------------------------
    //Creates a new list.
    //-----------------------------------------------------------------------------------------
    void New(int sz)    {size=sz; count=0; cursor=0;L=new T[size];L[0]=MIN;}
    //-----------------------------------------------------------------------------------------
    //Returns the index of key or 0 if the key is not in the list.
    //-----------------------------------------------------------------------------------------
    int Index(T key)
    {   int lo=1,hi=count;
        while (lo<=hi) {
            int m=(lo+hi)/2;
            if (key==L[m]) return m;
            if (key<L[m]) hi=m-1; else lo=m+1;
        }
        return 0;
    }
protected:
    const T MIN;    //Minimum value of type T
```

Figure 1. template class *List* (array implementation)

```
public:
    //-----------------------------------------------------------------------------------------
    /Create an empty list.
    //-----------------------------------------------------------------------------------------
    void Empty(void){int sz=size; if (L) delete[] L; New(sz);}
    //-----------------------------------------------------------------------------------------
    //Constructor
    //-----------------------------------------------------------------------------------------
    List(T m,int sz=100):size(sz),count(0),cursor(0),MIN(m){L=new T[size];L[0]=MIN;}
    //-----------------------------------------------------------------------------------------
    //Constructor. Read values of type T into the list from stream i.
    //-----------------------------------------------------------------------------------------
    List(T m,istream& i,int sz=100):size(sz),count(0),cursor(0),MIN(m)
    {L=new T[size];L[0]=MIN;Scan(i);}
    //-----------------------------------------------------------------------------------------
    //Destructor, reclaim storage.
    //-----------------------------------------------------------------------------------------
    ~List(){if (L) delete[] L;}
    //-----------------------------------------------------------------------------------------
    //Determine if the List is empty
    //-----------------------------------------------------------------------------------------
    bool IsEmpty(void){return count<=0;}
    //-----------------------------------------------------------------------------------------
    //Determine if the List is full
    //-----------------------------------------------------------------------------------------
    bool IsFull(void){return count>size-1;}
    //-----------------------------------------------------------------------------------------
    //Insert an element on the List
    //-----------------------------------------------------------------------------------------
    void Insert(T key)
    {   if (IsMember(key)) return;
        if (IsFull()) throw ListException("full");
        int i=++count;
        for (;key<L[i-1];i--) L[i]=L[i-1];
        L[i]=key;
    }
    //-----------------------------------------------------------------------------------------
    //Remove an element from the List
    //-----------------------------------------------------------------------------------------
    void Remove(T key)
    {   int i=Index(key);
        if (i==0) return;
        for (;i<count;i++)L[i]=L[i+1];
        count--;
    }
```

Figure 1. template class *List* (array implementation) continued

```
    //-----------------------------------------------------------------------------------------
    //Print the list L={v1,v2,...,vn}
    //-----------------------------------------------------------------------------------------
    void Print(ostream& o,char* title)
    {   o << endl << title << "{";
        for (int a=1;a<=count;a++) {
            if (a>1) o << ",";
            o << L[a];
        }
        o << "}"<< endl;
    }
    //-----------------------------------------------------------------------------------------
    //Read elements stored in stream i into the list.
    //-----------------------------------------------------------------------------------------
    void Scan(istream& i)
    {   for (;;) {
            T key;
            i >> key;
            if (i.eof()) break;
            Insert(key);
        }
    }
    //-----------------------------------------------------------------------------------------
    //Assign the index of the smallest element to the cursor.
    //-----------------------------------------------------------------------------------------
    void First(void){cursor=1;}
    //-----------------------------------------------------------------------------------------
    //Move the cursor to the next larger element. Stop when the index
    //of the cursor is just beyond the largest element.
    //-----------------------------------------------------------------------------------------
    void Next(void){if (cursor<count+1) cursor++;}
    //-----------------------------------------------------------------------------------------
    //Determine if the cursor is at the end of the list.
    //-----------------------------------------------------------------------------------------
    bool IsEol(void){return cursor>count;}
    //-----------------------------------------------------------------------------------------
    //Return the value of the key stored in the current element.
    //-----------------------------------------------------------------------------------------
    T Member(void){if (cursor) return L[cursor];}
    //-----------------------------------------------------------------------------------------
    //Determine if the key is a member of the list.
    //-----------------------------------------------------------------------------------------
    bool IsMember(T key){return Index(key);}
};
```

Figure 1. template class *List* (array implementation) continued