```
class List                  // specify invariants
{    ...                    // private data members
 public:
   List(int size = DEFAULT_CAPACITY);
   List(List&);
   ~List();
   List& operator=(const List&);
   …
   int           Count() const;
   Item&         Get(int index) const;
   Item&         First() const;
   Item&         Last() const;
   bool          Includes(const Item&) const;

   // destructive operations: change state
   void  Append(const Item&);
   void  Prepend(const Item&);
   void  Remove(const Item&);
   void  RemoveLast();
   void  RemoveFirst();
   void  RemoveAll();
};                 // FIRST, must determine characteristics of the class
```

Interface Invariants (Application Programmer)

- Minimal: illegal calls (unspecified behavior)
  - Cannot add beyond capacity
  - Cannot get() if index out of range
  - Cannot query() if List empty
- Problematic: default behavior may be expected
  - Cannot alter if List empty
  - Constructor must supply a positive integer
- Unnecessary: condition enforced by compiler
  - Valid type (Item&) passed

Implementation Invariants (Class Designer/Modifier)

- Minimal: implied by Interface invariants
  - No default behavior for accessors
  - Nop if remove invoked for empty List
  - Nop if remove invoked with invalid entry
- Problematic: of questionable validity
  - LL will not run out of memory
  - Array implementation is ordered
- Unnecessary: implied by function prototype
  - first(), last() non-destructive (implied by const)

Preconditions

- Minimal: implied by above invariants
  - get() has valid index
  - first(), last() called only on non-empty List
  - prepend(), append() called only on non-full List
- Problematic: of questionable validity
  - Includes() cannot be called with empty List
- Unnecessary: implied by function prototype
  - prepend(), append() called with valid Item

PostConditions

- Minimal: describes STATE after operation done
  - List may be empty after first(), last(), remove
  - List empty after removeAll()
  - List non-empty after append(), prepend()
- Problematic: of questionable validity
  - DEFAULT_CAPACITY is public
- Unnecessary: describes what functions does
  - List object exists after List()
  - One fewer item in List after remove

Class Invariants

- Minimal: implied by above invariants
  - List object holds between 0 and size items
- Problematic: of questionable validity
  - List objects do not contain duplicate values
- Unnecessary: implied by functions provided
  - List objects are not unique
  - Copying supported