Linked List ADT

• A linked list is:
  1. a linear data structure
  2. a data structure where each node has a unique predecessor and a unique successor

• A data element can be inserted or removed anywhere in the list
Linked List ADT Specification

- **Elements**: List elements can be of any type, but we will assume ListElement

- **Structure**: Any mechanism for allowing the insertion, deletion, or modification of a ListElement anywhere in the list. Each ListElement has a unique predecessor and successor
Linked List ADT Continued

- **Domain**: The number of list elements is bounded. A list is considered full if the upper-bound is reached. A list with no elements is considered empty.

- Operations: There are 18 operations.
Linked List Operations

Allocation and Deallocation
1. lstCreate
2. lstDispose

Checking number of elements
3. lstSize
4. lstIsFull
5. lstIsEmpty
Linked List Operations

- **Peek Operations**
  6. lstPeek
  7. lstPeekPrev
  8. lstPeekNext

- **Retrieving values**
  9. lstFirst
  10. lstLast
  11. lstNext
  12. lstPrev
List Operations

- Retrieving values
  13. lstDeleteCurrent
  14. lstInsertAfter
  15. lstInsertBefore
  16. lstUpdateCurrent
  17. lstHasNext
  18. lstHasPrev
Linked Lists

Singly Linked List

Singly Linked Circular List
Linked Lists

Doubly Linked List

Doubly Linked Circular List
Linked List Implementation

• How might we implement the previously specified Linked List ADT?
Implementation

- There are any number of ways but let’s begin with the following:

```c
typedef int DATATYPE;

typedef struct ListElement *ListElementPtr;
typedef struct ListElement
{
    DATATYPE data;
    ListElementPtr psNext;
} ListElement;

typedef struct List *ListPtr;
typedef struct List
{
    ListElementPtr psHead;
    int numElements;
} List;
```