Representing Lists With Pointers

**Idea:** Associate with each vertex *pointers* to the addresses in memory of the previous and next elements in the list.
• A pointer that has value 323 does not denote the 323rd element of the list (except possibly by pure chance).

• When following next pointers, one is in general skipping around between arbitrary positions in memory.

• Headers are often not used because they can make it more cumbersome to manipulate lists; in this case, global information about the list must be explicitly stored.

• Memory must be allocated when creating new vertices (or headers) and in many applications should be reclaimed when they are no longer needed.
Singly Linked Lists

• Each list vertex has a data field and next field, but no field containing a pointer to the previous vertex.

• PREV(v, L) can now be expensive (e.g., if no information about the preceding vertex has been explicitly saved).

• However, most other operations are unaffected or can be easily modified to still work in $O(1)$ time. For example, DELETE(v, L) can be redefined to delete the vertex that follows v.