Linked Lists - Introduction
Dynamic Arrays Revisited

• Dynamic array can sometimes be slow
  – When?
  – Why?
Linked Lists - Characteristics

- Data elements held in structures called “links”
- Like a chain: each link is tied to the next

- Links are 1 – 1 with elements, allocated and released as necessary
- Each link points to next link in sequence, sometimes to previous link
- *Not contiguously stored!!!*
struct Link {
    /* Single link. */
    TYPE val;  /* Data contained by this link. */
    struct Link *next; /* Pointer to next link. */
};
All linked lists consist of links ... but there are other design decisions:

- Header *(special value to point to start)* or no header?
- Use null as terminator, or special value *(sentinel)* for end?
- Use single or double links?
- Pointer to first element only, or pointer to first and last?
Implementing a stack interface with a linked list:

- Header with head reference only: null if empty
- Null terminated
- Singly linked
- Where should the ‘top’ of the stack be????
- Only access first element
struct linkedListStack {
    struct Link *firstLink; /* Initialize routine sets to zero/NULL. */
};

void linkedListStackInit (structlinkedListStack s) {
    s->firstLink = 0;
}
void pushListStack(struct ListStack *s, TYPE d) {
    /* You are going to write this:
        1. Allocate (malloc) a new link (check that it works!).
        2. Set data fields in the new link.
        3. Change head to point to new link. */
}

Linked List Tips...

• Draw the diagram!
• Go through the steps visually, labeling each step
• Convert each step to C code
• Try the boundary cases:
  – Empty list?
  – List with several items?
Other Linked List Operations

• How do you tell if stack is empty?

• How do you return first element (i.e., firstLink)?

• How do you remove an element?
Your Turn

• Complete Worksheet 17: Linked List Introduction, List Stack