```c
// ------------------------------------------------------
// Collected functions: does not compile
//
// Shows the difference of interfaces:
//
//         functional/procedural
//         recursvie/iterative
// ------------------------------------------------------

// Interface: recursive + functional
// Single return statement
//
// Call like this: anchor = list_remove_tail(anchor);
//
// Remove tail node of list
node_t* list_remove_tail(node_t* node){

  node_t* result;
  if (node == NULL) {
    result = NULL;
  } else {
    if (node->next == NULL) {
      // Remove last node
      free(node);
      result = NULL;
    } else {
      // Recursive call
      node->next = list_remove_tail(node->next);
      result = node;
    }
  }

  return result;
}
```

```
// ------------------------------------------------------

// Interface: recursive + procedural
// Single return statement
//
// Call like this: list_remove_tail(&anchor);
//
// Remove tail node of list
void list_remove_tail(node_t* *pnode){
  node_t* node = *pnode;
  node_t* result;
  if (node == NULL) {
    result = NULL;
  } else {
    if (node->next == NULL) {
      // Remove last node
      free(node);
      result = NULL;
    } else {
      // Recursive call
      list_remove_tail(&node->next);
      result = node;
    }
  }
  *pnode = result;
  return;
}
```

```c
// ------------------------------------------------------

// Interface: iterative + functional
// Single return statement
//
// Call like this: anchor = list_remove_tail(anchor);
//
// Remove tail node of list
node_t* list_remove_tail(node_t* node){

  node_t *result = node; // init result
  if (node != NULL) {
    node_t* last = NULL;   // Last node before tail node

    // Go to tail node in list
    while (node->next != NULL) {
      last = node;
      // Advance iteration
      node = node -> next;
    }

    // node points to tail node in list
    // Free that node
    free(node);

    if (last == NULL) {
      // node was the only node in the list
      result = NULL;
    } else {
      // List had at least two elements
      // Terminate list
      last->next = NULL;
    }
  }

  return result;
}
```

```c
// ------------------------------------------------------

// Interface: iterative + procedural
// Single return statement
//
// Call like this: list_remove_tail(&anchor);
//
// Remove tail node of list
void list_remove_tail(node_t* *pnode){
  node_t *node = *pnode;
  node_t *result = node; // init result
  if (node != NULL) {
    node_t* last = NULL;   // Last node before tail node

    // Go to tail node in list
    while (node->next != NULL) {
      last = node;
      // Advance iteration
      node = node -> next;
    }

    // node points to tail node in list
    // Free that node
    free(node);

    if (last == NULL) {
      // node was the only node in the list
      result = NULL;
    } else {
      // List had at least two elements
      // Terminate list
      last->next = NULL;
    }
  }
  *pnode = result;
  return;
}
```