Name: Id: section:

1. Which pointer update order prevents losing the rest of the list when inserting newNode after curr?

- **A. curr->next = newNode; newNode->next = curr->next;**
- **B. newNode->next = curr->next; curr->next = newNode;**
- **C. newNode->next = curr; curr->next = newNode;**
- **D. curr = curr->next; newNode->next = curr;**

2. Given head may be null, which condition is safest before dereferencing head->next?

- **A. if (head->next)**
- **B. if (!head)**
- **C. if (head && head->next)**
- **D. if (head || head->next)**

3. What is the safest way to delete the head node when head may be null?

- **A. delete head; head = head->next;**
- **B. Node* tmp = head; head = head->next; delete tmp;**
- **C. head = head->next; delete head;**
- **D. if (head) { Node* tmp = head; head = head->next; delete tmp; }**

4. Consider deleting the node AFTER curr. Which is correct?

- **A. Node* del = curr->next; curr->next = del->next; delete del;**
- **B. delete curr->next; curr->next = curr->next->next;**
- **C. curr->next = curr->next->next; delete curr->next;**
- **D. Node* del = curr; curr->next = del->next; delete del;**

5. Traversal to print all elements is typically:

- **A. While pointer not null, print, move pointer**
- **B. For i in [0..n): print head->val**
- **C. Recursively print only if node->next**
- **D. Do-while until node == head**