1. Write a function that takes two sorted linked lists as input and merges them into a single sorted linked list. Consider the following and complete the missing code.

|  |
| --- |
| typedef struct Node {  element val;  struct Node\* next;  } Node;  // Function to merge two sorted linked lists  Node\* mergeTwoLists(Node\* l1, Node\* l2) {  **//complete code here**  }  int main() {  // Create two sorted linked lists  Node\* l1 = (Node\*) malloc(sizeof(Node));  l1->val = 1;  Node\* l1\_node2 = (Node\*) malloc(sizeof(Node));  l1\_node2->val = 2;  l1->next = l1\_node2;  l1\_node2->next = NULL;  Node\* l2 = (Node\*) malloc(sizeof(Node));  l2->val = 1;  Node\* l2\_node2 = (Node\*) malloc(sizeof(Node));  l2\_node2->val = 3;  l2->next = l2\_node2;  l2\_node2->next = NULL;  // Merge the two lists and print the merged list  Node\* merged\_list = mergeTwoLists(l1, l2);  while (merged\_list != NULL) {  printf("%d -> ", merged\_list->val);  merged\_list = merged\_list->next;  }  printf("NULL\n");  return 0;  } |

1. Implement a function that reverses a linked list in place, i.e., without creating a new linked list.

|  |
| --- |
| // Define the linked list node structure  typedef struct Node {  element val;  struct Node\* next;  } Node;  // Function to reverse a linked list in place  Node\* reverseList(Node\* head) {  // Complete code here  }  int main() {  // Create a sample linked list  Node\* head = (Node\*) malloc(sizeof(Node));  head->val = 1;  Node\* node2 = (Node\*) malloc(sizeof(Node));  node2->val = 2;  Node\* node3 = (Node\*) malloc(sizeof(Node));  node3->val = 3;  Node\* node4 = (Node\*) malloc(sizeof(Node));  node4->val = 4;  head->next = node2;  node2->next = node3;  node3->next = node4;  node4->next = NULL;  // Reverse the list and print the reversed list  Node\* reversed\_list = reverseList(head);  while (reversed\_list != NULL) {  printf("%d -> ", reversed\_list->val);  reversed\_list = reversed\_list->next;  }  printf("NULL\n");  return 0;  } |

1. Write a function that removes all duplicates from a sorted linked list in C. Complete the code below.

|  |
| --- |
| // Define the linked list node structure  typedef struct Node {  element val;  struct Node\* next;  } Node;  // Function to remove duplicates from a linked list  void removeDuplicates(Node\* head) {  **// Complete code here**  }  int main() {  // Create a sample linked list  Node\* head = (Node\*) malloc(sizeof(Node));  head->val = 1;  Node\* node2 = (Node\*) malloc(sizeof(Node));  node2->val = 2;  Node\* node3 = (Node\*) malloc(sizeof(Node));  node3->val = 2;  Node\* node4 = (Node\*) malloc(sizeof(Node));  node4->val = 3;  Node\* node5 = (Node\*) malloc(sizeof(Node));  node5->val = 3;  Node\* node6 = (Node\*) malloc(sizeof(Node));  node6->val = 3;  head->next = node2;  node2->next = node3;  node3->next = node4;  node4->next = node5;  node5->next = node6;  node6->next = NULL;  // Remove duplicates and print the resulting list  removeDuplicates(head);  Node\* curr = head;  while (curr != NULL) {  printf("%d -> ", curr->val);  curr = curr->next;  }  printf("NULL\n");  return 0;  } |