Task 1

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node* next;
};


struct Node* createNode(int data) {
    struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}


void addNodeAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}


void addNodeAtEnd(struct Node** head, int data) {
```

```c
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}

void printLinkedList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;


    addNodeAtBeginning(&head, 5);
    addNodeAtEnd(&head, 10);
    addNodeAtEnd(&head, 15);


    printf("Linked List: ");
    printLinkedList(head);
    return 0;
}
```

Task 2

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void addNodeAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}

void addNodeAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}

void insertAfterValue(struct Node* head, int
valueToInsertAfter, int data) {
    struct Node* newNode = createNode(data);
    struct Node* current = head;
    while (current != NULL) {
        if (current->data == valueToInsertAfter) {
            newNode->next = current->next;
            current->next = newNode;
            return;
        }
        current = current->next;
    }
}

void deleteByValue(struct Node** head, int valueToDelete) {
    if (*head == NULL) {
        return;
    }

    if ((*head)->data == valueToDelete) {
        struct Node* temp = *head;
        *head = (*head)->next;
        free(temp);
        return;
    }

    struct Node* current = *head;
    while (current->next != NULL) {
        if (current->next->data == valueToDelete) {
            struct Node* temp = current->next;
            current->next = current->next->next;
            free(temp);
            return;
        }
        current = current->next;
    }
}

void printLinkedList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;


    addNodeAtBeginning(&head, 5);
    addNodeAtEnd(&head, 10);
    insertAfterValue(head, 10, 25);


    deleteByValue(&head, 10);


    insertAfterValue(head, 5, 20);


    printf("Linked List: ");
    printLinkedList(head);

    return 0;
}
```

Task 3

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void addNodeAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}

void addNodeAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}

void reverseLinkedList(struct Node** head) {
    struct Node* prev = NULL;
    struct Node* current = *head;
    struct Node* next = NULL;

    while (current != NULL) {
        next = current->next; // Store the next node
        current->next = prev; // Change the current node's
next to the previous node
        prev = current;      // Move prev to the current
node
        current = next;      // Move current to the next
node
    }

    *head = prev; // Update the head to point to the new
first node (previously the last node)
}

void printLinkedList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;

    // Adding nodes to the beginning and end
    addNodeAtBeginning(&head, 5);
    addNodeAtEnd(&head, 25);
    addNodeAtEnd(&head, 20);

    // Printing the original linked list
    printf("Original Linked List: ");
    printLinkedList(head);

    // Reverse the linked list
    reverseLinkedList(&head);
    // Printing the reversed linked list
    printf("Reversed Linked List: ");
    printLinkedList(head);

    return 0;
}
```

Task 4

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void addNodeAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}

int hasCycle(struct Node* head, struct Node** cycleStart) {
    struct Node* slow = head;
    struct Node* fast = head;

    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;

        if (slow == fast) {

            *cycleStart = head;
            while (*cycleStart != slow) {
                *cycleStart = (*cycleStart)->next;
                slow = slow->next;
            }
            return 1;
        }
    }


    return 0;
}

int main() {
    struct Node* head = NULL;
    addNodeAtEnd(&head, 5);
    addNodeAtEnd(&head, 10);
    struct Node* cycleStartNode = head;
    addNodeAtEnd(&head, 15);
    addNodeAtEnd(&head, 20);
    struct Node* cycleEndNode = head;
    cycleEndNode->next = cycleStartNode;

    struct Node* cycleStart = NULL;
    int hasCycleResult = hasCycle(head, &cycleStart);

    printf("Has Cycle: %s\n", hasCycleResult ? "Yes" :
"No");
    if (hasCycleResult) {
        printf("Cycle Start Node: %d\n", cycleStart->data);
    }

    return 0;
}
```

Task 5

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void addNodeAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}

struct Node* mergeSortedLists(struct Node* listA, struct
Node* listB) {
    struct Node dummyNode;
    struct Node* tail = &dummyNode;
    dummyNode.next = NULL;

    while (1) {
        if (listA == NULL) {
            tail->next = listB;
            break;
```

```c
        }
        if (listB == NULL) {
            tail->next = listA;
            break;
        }

        if (listA->data <= listB->data) {
            tail->next = listA;
            listA = listA->next;
        } else {
            tail->next = listB;
            listB = listB->next;
        }
        tail = tail->next;
    }

    return dummyNode.next;
}

void printLinkedList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* listA = NULL;
    struct Node* listB = NULL;
    addNodeAtEnd(&listA, 5);
    addNodeAtEnd(&listA, 10);

    addNodeAtEnd(&listB, 7);
    addNodeAtEnd(&listB, 12);


    struct Node* mergedList = mergeSortedLists(listA,
listB);


    printf("Merged List: ");
    printLinkedList(mergedList);

    return 0;
}
```