Task1

```
#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));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

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

void addToEnd(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;
```

```c
}

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

int main() {
    struct Node* head = NULL;
    head = addToBeginning(head, 5);
    addToEnd(head, 10);
    addToEnd(head, 15);
    printf("Linked List: ");
    printList(head);
    return 0;
}
```

Task2

```
#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));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

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

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


struct Node* insertAtPosition(struct Node* head, int position, int data)
{
    struct Node* newNode = createNode(data);
    if (position == 0) {
        newNode->next = head;
        return newNode;
    }
    struct Node* current = head;
    int index = 0;
    while (current != NULL && index < position - 1) {
        current = current->next;
        index++;
    }
    if (current == NULL) {
        return head;
    }
    newNode->next = current->next;
    current->next = newNode;
    return head;
}

void deleteNodeAtPosition(struct Node* head, int position) {
    if (position == 0) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
        return;
    }
    struct Node* current = head;
    int index = 0;
    while (current != NULL && index < position - 1) {
        current = current->next;
        index++;
    }
    if (current == NULL || current->next == NULL) {
        return;
    }
    struct Node* temp = current->next;
    current->next = temp->next;
    free(temp);
```

```c
}

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

int main() {
    struct Node* head = NULL;
    head = addToBeginning(head, 5);
    addToEnd(head, 10);
    addToEnd(head, 15);

    head = insertAfterValue(head, 10, 25);
    deleteNodeByValue(head, 10);
    head = insertAtPosition(head, 2, 20);
    deleteNodeAtPosition(head, 3);

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

    return 0;
}
```

Task3

}

```c
#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));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
struct Node* insert(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = head;
    return newNode;
}

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

```c
struct Node* reverseList(struct Node* head) {
    struct Node* prev = NULL;
    struct Node* current = head;
    struct Node* next = NULL;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}

int main() {
    struct Node* head = NULL;
    head = insert(head, 5);
    head = insert(head, 25);
    head = insert(head, 20);
    printf("Original: ");
    printList(head);
    head = reverseList(head);
    printf("Reversed: ");
    printList(head);

    return 0;
}
```

Task4

```c
#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));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void addNode(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    *head = 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) {
            slow = head;
            while (slow != fast) {
                slow = slow->next;
                fast = fast->next;
            } *cycleStart = slow;
            return 1;
        }
    }
    return 0;
}
int main() {
    struct Node* head = NULL;
    struct Node* cycleStart = NULL;
    addNode(&head, 10);
    addNode(&head, 20);
    addNode(&head, 30);
    addNode(&head, 40);
    addNode(&head, 50);
    struct Node* node50 = head;
    while (node50->next != NULL) {
        node50 = node50->next;
    }
    struct Node* node10 = head;
    while (node10->next != NULL) {
        node10 = node10->next;
    }
    node50->next = node10;
    int result = hasCycle(head, &cycleStart);
    if (result) {
        printf("Has Cycle: Yes\n");
        printf("Cycle Start Node: %d\n", cycleStart->data);
    } else {
        printf("Has Cycle: No\n");
    }
    return 0;
}
```

Task5

```
#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));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

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


struct Node* mergeSortedLists(struct Node* list1, struct Node* list2) {
    struct Node* mergedList = NULL;
    while (list1 != NULL && list2 != NULL) {
```

```c
        if (list1->data < list2->data) {
            append(&mergedList, list1->data);
            list1 = list1->next;
        } else {
            append(&mergedList, list2->data);
            list2 = list2->next;
        }
    }
    while (list1 != NULL) {
        append(&mergedList, list1->data);
        list1 = list1->next;
    }
    while (list2 != NULL) {
        append(&mergedList, list2->data);
        list2 = list2->next;
    }

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



int main() {
    struct Node* listA = NULL;
    struct Node* listB = NULL;

    append(&listA, 5);
    append(&listA, 10);

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

    printf("List A: ");
    printList(listA);
    printf("List B: ");
    printList(listB);

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

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

    return 0;
}
```