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));
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;
}
struct Node *addToEnd(struct Node *head, int data) {
struct Node *newNode = createNode(data);
if (head == NULL) {
return newNode;
}
struct Node *current = head;
while (current->next != NULL) {
current = current->next;
}
current->next = newNode;
return head;
}
void printLinkedList(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);
head = addToEnd(head, 10);
head = addToEnd(head, 15);
printf("Linked List: ");
printLinkedList(head);
return 0;
}
```

Task 2:

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));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

struct Node* reverseLinkedList(struct Node* head) {

struct Node *prev = NULL, *current = head, *next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

return prev;

}

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 = createNode(5);
head->next = createNode(25);
head->next->next = createNode(20);
printf("Original Linked List: ");
printLinkedList(head);
head = reverseLinkedList(head);
printf("Reversed Linked List: ");
printLinkedList(head);
return 0;
}
```

<u>Task 4:</u>

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.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;
}
```

```
bool hasCycle(struct Node* head, int* cycleStartNode) {
if (head == NULL || head->next == NULL) {
return false;
}
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;
}
*cycleStartNode = slow->data;
return true;
}
}
return false;
}
int main() {
struct Node* head = createNode(10);
head->next = createNode(20);
head->next->next = createNode(30);
head->next->next->next = createNode(40);
head->next->next->next->next = head;
int cycleStartNode;
if (hasCycle(head, &cycleStartNode)) {
```

```
printf("Has Cycle: Yes\n");
printf("Cycle Start Node: %d\n", cycleStartNode);
} else {
printf("Has Cycle: No\n");
}
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));
newNode->data = data;
newNode->next = NULL;
return newNode;
}
struct Node* mergeSortedLists(struct Node* listA, struct Node* listB) {
struct Node* mergedList = NULL;
struct Node* tail = mergedList;
while (1) {
if (listA == NULL) {
tail->next = listB;
break;
}
if (listB == NULL) {
tail->next = listA;
break;
}
if (listA->data <= listB->data) {
if (mergedList == NULL) {
mergedList = tail = listA;
} else {
tail->next = listA;
tail = listA;
}
listA = listA->next;
} else {
if (mergedList == NULL) {
mergedList = tail = listB;
} else {
tail->next = listB;
tail = listB;
}
listB = listB->next;
}
}
return mergedList;
}
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 = createNode(5);
listA->next = createNode(10);
struct Node* listB = createNode(7);
listB->next = createNode(12);
printf("List A: ");
printLinkedList(listA);
printf("List B: ");
printLinkedList(listB);
struct Node* mergedList = mergeSortedLists(listA, listB);
printf("Merged List: ");
printLinkedList(mergedList);
return 0;
}
```