

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  struct Node {
4      int data;
5      struct Node* next;
6  };
7  struct Node *createNode(int data) {
8      struct Node *newNode = malloc(sizeof(struct Node));
9      if (newNode == NULL) {
10         printf("Memory allocation error.\n");
11         exit(EXIT_FAILURE);
12     }
13     newNode->data = data;
14     newNode->next = NULL;
15     return newNode;
16 }
17 struct Node* merge_list(struct Node* head1, struct Node* head2) {
18     struct Node *head3 = NULL;
19     struct Node *current1 = head1;
20     struct Node *current2 = head2;
21
22     while (current1 != NULL && current2 != NULL) {
23         struct Node* newNode = createNode(0);
24
25         if (current1->data < current2->data) {
26             newNode->data = current1->data;
27             current1 = current1->next;
28         } else if (current1->data > current2->data) {
29             newNode->data = current2->data;
30             current2 = current2->next;
31         } else if (current1->data == current2->data) {
32             newNode->data = current1->data;
33             current1 = current1->next;
34             current2 = current2->next;
35         }
36
37         if (head3 == NULL) {
38             head3 = newNode;
39         } else {
40             struct Node* current3 = head3;
41             while (current3->next != NULL) {
42                 current3 = current3->next;
43             }
44             current3->next = newNode;
45         }
46     }
47     while (current1 != NULL) {
48         struct Node* newNode = createNode(current1->data);
49         current1 = current1->next;
50
51         if (head3 == NULL) {
52             head3 = newNode;
53         } else {
54             struct Node* current3 = head3;
55             while (current3->next != NULL) {
56                 current3 = current3->next;
57             }
58             current3->next = newNode;
59         }
60     }
61     while (current2 != NULL) {
62         struct Node* newNode = createNode(current2->data);
63         current2 = current2->next;
64         if (head3 == NULL) {
65             head3 = newNode;
66         } else {

```

```

67         struct Node *current3 = head3;
68         while (current3->next != NULL) {
69             current3 = current3->next;
70         }
71         current3->next = newNode;
72     }
73 }
74
75 return head3;
76 }
77 void printList(struct Node* head) {
78     while (head != NULL) {
79         printf("%d -> ", head->data);
80         head = head->next;
81     }
82     printf("NULL\n");
83 }
84 void freeList(struct Node* head) {
85     struct Node* temp;
86     while (head != NULL) {
87         temp = head;
88         head = head->next;
89         free(temp);
90     }
91 }
92
93 int main() {
94     struct Node* list1 = createNode(1);
95     list1->next = createNode(3);
96     list1->next->next = createNode(5);
97     struct Node* list2 = createNode(2);
98     list2->next = createNode(4);
99     list2->next->next = createNode(6);
100    printf("Original List 1: ");
101    printList(list1);
102    printf("Original List 2: ");
103    printList(list2);
104    struct Node* mergedList = merge_list(list1, list2);
105    printf("Merged List: ");
106    printList(mergedList);
107    freeList(list1);
108    freeList(list2);
109    freeList(mergedList);
110
111    return 0;
112

```