1. <u>Merge two sorted linked lists</u>

Create a method to merge two singly linked lists into one sorted linked list. The merged list should maintain the sorted order of elements.

To perform the above operation, both the list should be traversed, comparing the elements at each step, and adding the smaller element to the merged list. If one of the lists is exhausted before the other, we simply append the remaining elements of the other list to the merged list.

Program Code:

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

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

typedef struct slinklist node;

node *start1 = NULL;
node *start2 = NULL;

int menu() {
   int ch;
   printf("\n 1. Create lists");
   printf("\n-------------------------");
   printf("\n 2. Insert a node into list 1");
   printf("\n-------------------------");
   printf("\n 3. Insert a node into list 2");
   printf("\n-------------------------");
   printf("\n 4. Merge two lists");
   printf("\n-------------------------");
   printf("\n 5. Exit ");
   printf("\n\n Enter your choice: ");
   scanf("%d", &ch);
   return ch;
}

node* getnode() {
   node *newnode;
   newnode = (node *)malloc(sizeof(node));
   printf("\n Enter data: ");
   scanf("%d", &newnode->data);
   newnode->next = NULL;
   return newnode;
}

void createlist(node **start, int n) {
   int i;
   node *newnode;
```

```
    node *temp;
    for (i = 0; i < n; i++) {
        newnode = getnode();
        if (*start == NULL) {
            *start = newnode;
        } else {
            temp = *start;
            while (temp->next != NULL)
                temp = temp->next;
            temp->next = newnode;
        }
    }
}

void display(node *start) {
    node *temp;
    temp = start;
    printf("\n The contents of the list (Left to Right): \n");
    if (start == NULL) {
        printf("\n Empty List");
        return;
    } else {
        while (temp != NULL) {
            printf("%d-->", temp->data);
            temp = temp->next;
        }
    }
    printf(" X ");
}

void insert_node(node **start) {
    node *newnode = getnode();
    if (*start == NULL) {
        *start = newnode;
    } else {
        node *temp = *start;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
    }
    printf("\n Node inserted successfully.\n");
    display(*start);
}

// Complete merge_list function below

node* merge_lists(node *list1, node *list2) {
    node dummy;
    dummy.next = NULL;
    node *tail = &dummy;
```

*Write your code here

```
    return dummy.next;
}

void main(void) {
    int ch, n;
    while (1) {
        ch = menu();
        switch (ch) {
            case 1:
                if (start1 == NULL && start2 == NULL) {
                    printf("\n Number of nodes you want to create in list 1: ");
                    scanf("%d", &n);
                    createlist(&start1, n);
                    printf("\n List 1 created..");
                    display(start1);

                    printf("\n\n Number of nodes you want to create in list 2: ");
                    scanf("%d", &n);
                    createlist(&start2, n);
                    printf("\n List 2 created..");
                    display(start2);
                } else {
                    printf("\n Lists are already created..");
                }
                break;
            case 2:
                insert_node(&start1);
                break;
            case 3:
                insert_node(&start2);
                break;
            case 4:
                if (start1 == NULL || start2 == NULL) {
                    printf("\n One or both lists are empty, cannot merge\n");
                } else {
                    node *merged_list = merge_lists(start1, start2);
                    printf("\n Merged List: ");
                    display(merged_list);
                }
                break;
            default:
                exit(0);
        }
    }
}
```