```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 add(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void sort(struct Node** head) {
    struct Node *i, *j;
    int temp;
    for (i = *head; i != NULL; i = i->next) {
        for (j = i->next; j != NULL; j = j->next) {
            if (i->data > j->data) {
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }
        }
    }
}

void reverse(struct Node** head) {
    struct Node *prev = NULL, *current = *head, *next = NULL;
    while (current != NULL) {
```

```
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head = prev;
}

void concatenate(struct Node* L1, struct Node* L2) {
    if (L1 == NULL) {
        L1 = L2;
        return;
    }
    struct Node* temp = L1;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = L2;
}

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

int main() {
    struct Node* L1 = NULL;
    struct Node* L2 = NULL;
    int c, d;
    printf("Enter your choice:\n 1. Append to L1\n 2. Append to L2\n 3. Sort L1\n
4. Sort L2\n 5. Display L1\n 6. Reverse L1\n 7. Concatenate L2 to L1\n 8.
Exit\n");

    while (1) {
        printf("Enter choice: ");
        scanf("%d", &c);
        switch (c) {
            case 1:
                printf("Enter value: ");
                scanf("%d", &d);
                add(&L1, d);
                break;
```

```
            case 2:
                printf("Enter value: ");
                scanf("%d", &d);
                add(&L2, d);
                break;
            case 3:
                sort(&L1);
                break;
            case 4:
                sort(&L2);
                break;
            case 5:
                display(L1);
                break;
            case 6:
                reverse(&L1);
                break;
            case 7:
                concatenate(L1, L2);
                break;
            case 8:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}
```

Output:

```
Enter your choice:
 1. Append to L1
 2. Append to L2
 3. Sort L1
 4. Sort L2
 5. Display L1
 6. Reverse L1
 7. Concatenate L2 to L1
 8. Exit
Enter choice: 2
Enter value: 2
Enter choice: 1
Enter value: 2
Enter choice: 1
Enter value: 1
Enter choice: 1
Enter value: 8
Enter choice: 1
Enter value: 2
Enter choice: 2
Enter value: 3
Enter choice: 2
Enter value: 5
Enter choice: 5
2 1 8 2
Enter choice: 3
Enter choice: 4
Enter choice: 5
1 2 2 8
Enter choice: 6
Enter choice: 5
8 2 2 1
Enter choice: 7
Enter choice: 5
8 2 2 1 2 3 5
Enter choice:
```

PAGE :
DATE : / /

```
#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 add ( struct Node** head , int data)
{
  struct Node* newNode = createNode(data)
  if ( *head == NULL)
  {
     *head = newNode;
     return;
  }
  struct Node* temp = *head
  while (temp -> next != NULL)
  {
     temp = temp -> next;
  }
  temp -> next = newNode;
}
```

```
void sortL (struct Node** head)
{
   struct Node* prev = NULL, *current
   struct Node *i, *j;
   int temp;
   for ( i = *head ; i != NULL ; i = i->next)
    {
      for ( j = i->next ; j != NULL ; j = j->next)
       {
          if (i->data > j->data)
          {
             temp = i->data;
             i->data = j->data
             j->data = temp;
          }
       }
    }
}

void reverseL (struct Node **head)
{
   struct Node *prev = NULL , *current = *head , *next=NULL;
   while (current != NULL)
    {
       next = current->next;
       current->next = prev;
       prev = current
       current = next;
    }
   *head = prev;
}
```

```
void concatenate (struct Node* l1h, struct Node* l2h)
{
    if (l1h != NULL)
    {
        l1h = l2h;
        return;
    }
    struct Node* temp = l1h;
    while (temp -> next != NULL)
    {
        temp = temp -> next;
    }
    temp -> next = l2h;
}

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

int main()
{
    struct Node* l1h = NULL;
    struct Node* l2h = NULL;
    int c, d;
    printf("Enter \n 1. Append to l1 \n 2 Append to l2 \n 3.
        display l1 \n 4. display l2 \n 5. sort l1 \n
        6. Reverse l1 \n 7. concatenate l2 to l1 \n 8. Exit");
```

```
while(1)
{
  printf("Enter your choice");
  scanf("%d", &ch);

  switch(c)
  {
    case 1:
        printf("Enter data");
        scanf("%d", &data);
        append
        add(&llh, data);
        break;
    case 2:
        printf("Enter data");
        scanf("%d", &d)
        add(&l2h, data);
        break;
    case 3:
        printf("List 1: ");
        display(llh);
        break;
    case 4:
        printf("list 2: ");
        display(l2h);
        break;
    case 5:
        sort(&llh);
        break;
    case 6:
        reverse(llh);
    Ca  break;
    Case 7:
        concatenate(&llh, &l2h);
```

PAGE :
DATE : / /

```
case 8:
    return 0;

default:
    printf("Invalid input");
```

Output:

```
Enter Choice
1. Append to l1
2. Append to l2
3. display l1
4. display l2
5. sort l1
6. reverse l1
7. concatenate l2 to l1
8. Exit
Enter choice: 1
Enter data: 1
Enter choice: 1
Enter data: 5
Enter choice 1
Enter data: 0
Enter choice 2:
Enter data: 12
Enter choice 2
Enter data: 15
Enter choice 5
Enter choice 3
list1: 5 0, 1, 5
Enter choice 6
Enter choice 3
list1: 5, 1, 0
```

```
Enter choice 7
Enter choice 3
list1: 5, 1, 0, 12, 15
```