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

struct node {
    int info;
    struct node* link;
};

void createList(struct node** start);
void displayList(struct node* start);
void sort(struct node* start);
void reverseLL(struct node** start);
void concat(struct node** start1, struct node* start2);

int main() {
    struct node* start1 = NULL;
    struct node* start2 = NULL;
    int choice;

    while (1) {
        printf("1. Create List 1\n");
        printf("2. Create List 2\n");
        printf("3. Display List 1\n");
        printf("4. Display List 2\n");
        printf("5. Sort List 1\n");
        printf("6. Sort List 2\n");
        printf("7. Reverse List 1\n");
        printf("8. Reverse List 2\n");
        printf("9. Concatenate Lists\n");
        printf("0. Exit\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createList(&start1);
                break;

            case 2:
                createList(&start2);
                break;

            case 3:
                displayList(start1);
```

```
                break;

            case 4:
                displayList(start2);
                break;

            case 5:
                sort(start1);
                printf("List 1 sorted.\n");
                break;

            case 6:
                sort(start2);
                printf("List 2 sorted.\n");
                break;

            case 7:
                reverseLL(&start1);
                printf("List 1 reversed.\n");
                break;

            case 8:
                reverseLL(&start2);
                printf("List 2 reversed.\n");
                break;

            case 9:
                concat(&start1, start2);
                printf("Lists concatenated.\n");
                break;

            case 0:
                printf("Exiting from the program.\n");
                exit(0);

            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

```c
void createList(struct node** start) {
    if (*start == NULL) {
        int n, data;
        printf("\nEnter the number of nodes: ");
        scanf("%d", &n);

        struct node* newnode;
        struct node* temp;

        for (int i = 1; i <= n; i++) {
            newnode = malloc(sizeof(struct node));

            printf("\nEnter number to be inserted: ");
            scanf("%d", &data);

            newnode->info = data;
            newnode->link = NULL;

            if (*start == NULL) {
                *start = newnode;
                temp = newnode;
            } else {
                temp->link = newnode;
                temp = temp->link;
            }
        }
        printf("\nThe list is created\n");
    } else {
        printf("\nThe list is already created\n");
    }
}


void displayList(struct node* start) {
    if (start == NULL) {
        printf("List is empty.\n");
    } else {
        struct node* current = start;

        printf("List: ");
        while (current != NULL) {
            printf("%d -> ", current->info);
            current = current->link;
```

```
        }
        printf("NULL\n");
    }
}


void sort(struct node* start) {
    struct node *current, *index;
    int temp;

    if (start == NULL || start->link == NULL) {
        return;
    } else {
        current = start;

        while (current != NULL) {
            index = current->link;

            while (index != NULL) {
                if (current->info > index->info) {
                    temp = current->info;
                    current->info = index->info;
                    index->info = temp;
                }
                index = index->link;
            }

            current = current->link;
        }
    }
}


void reverseLL(struct node** start) {
    struct node *prev, *current, *next;
    current = (*start)->link;
    prev = NULL;

    while (current != NULL) {
        next = current->link;
        current->link = prev;
        prev = current;
        current = next;
```

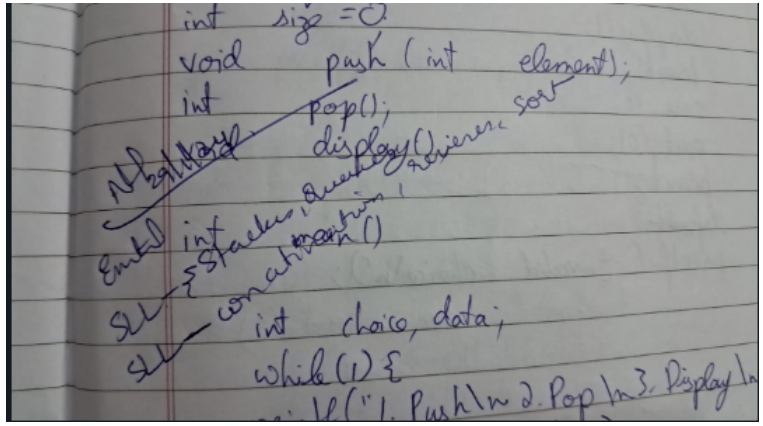
```
    }

    (*start)->link = prev;
}

void concat(struct node** start1, struct node* start2) {
    if (start2 == NULL) {
        return;  // Nothing to concatenate
    }

    struct node* temp = *start1;

    if (*start1 == NULL) {
        *start1 = start2;
    } else {
        while (temp->link != NULL) {
            temp = temp->link;
        }

        temp->link = start2;
    }
}
```

```
PS C:\Users\kadab\OneDrive\Desktop\DS> gcc nine.c
PS C:\Users\kadab\OneDrive\Desktop\DS> .\a.exe
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
1

Enter the number of nodes: 2

Enter number to be inserted: 1

Enter number to be inserted: 2

The list is created
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
2

Enter the number of nodes: 2

Enter number to be inserted: 5

Enter number to be inserted: 3
```

```
The list is created
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
5
List 1 sorted.
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
6
List 2 sorted.
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
7
```

```
List 1 reversed.
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
8
List 2 reversed.
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
9
Lists concatenated.
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
3
List: 1 -> 2 -> 3 -> 5 -> NULL
```

```
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
4
List: 3 -> 5 -> NULL
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Reverse List 1
8. Reverse List 2
9. Concatenate Lists
0. Exit
0
Exiting from the program.
PS C:\Users\kadab\OneDrive\Desktop\DS>
```


int size = 0;
void push (int element);
int pop();
display()
int
main()
int choice, data;
while(1) {