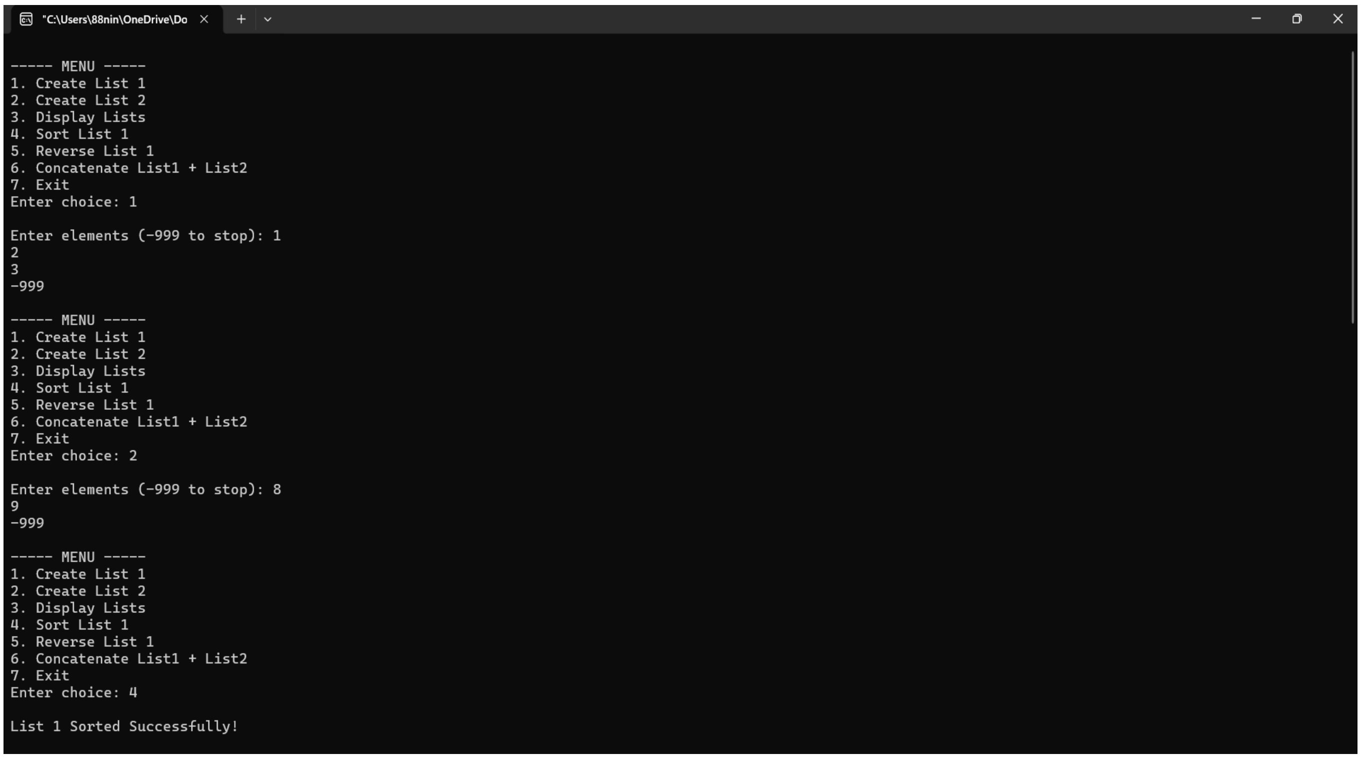
/* WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists. */

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

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

// Function prototypes
struct node* createList();
void display(struct node *start);
struct node* sortList(struct node *start);
struct node* reverseList(struct node *start);
struct node* concatenate(struct node *start1, struct node *start2);

int main() {
   struct node *list1 = NULL, *list2 = NULL;
   int choice;

   while (1) {
      printf("\n----- MENU -----\n");
      printf("1. Create List 1\n");
printf("2. Create List 2\n");
printf("3. Display Lists\n");
printf("4. Sort List 1\n");
printf("5. Reverse List 1\n");
printf("6. Concatenate List1 + List2\n");
printf("7. Exit\n");
printf("Enter choice: ");
scanf("%d", &choice);

switch (choice) {

case 1:
    list1 = createList();
    break;

case 2:
    list2 = createList();
    break;

case 3:
    printf("\nList 1: ");
    display(list1);
    printf("\nList 2: ");
    display(list2);
    break;
```

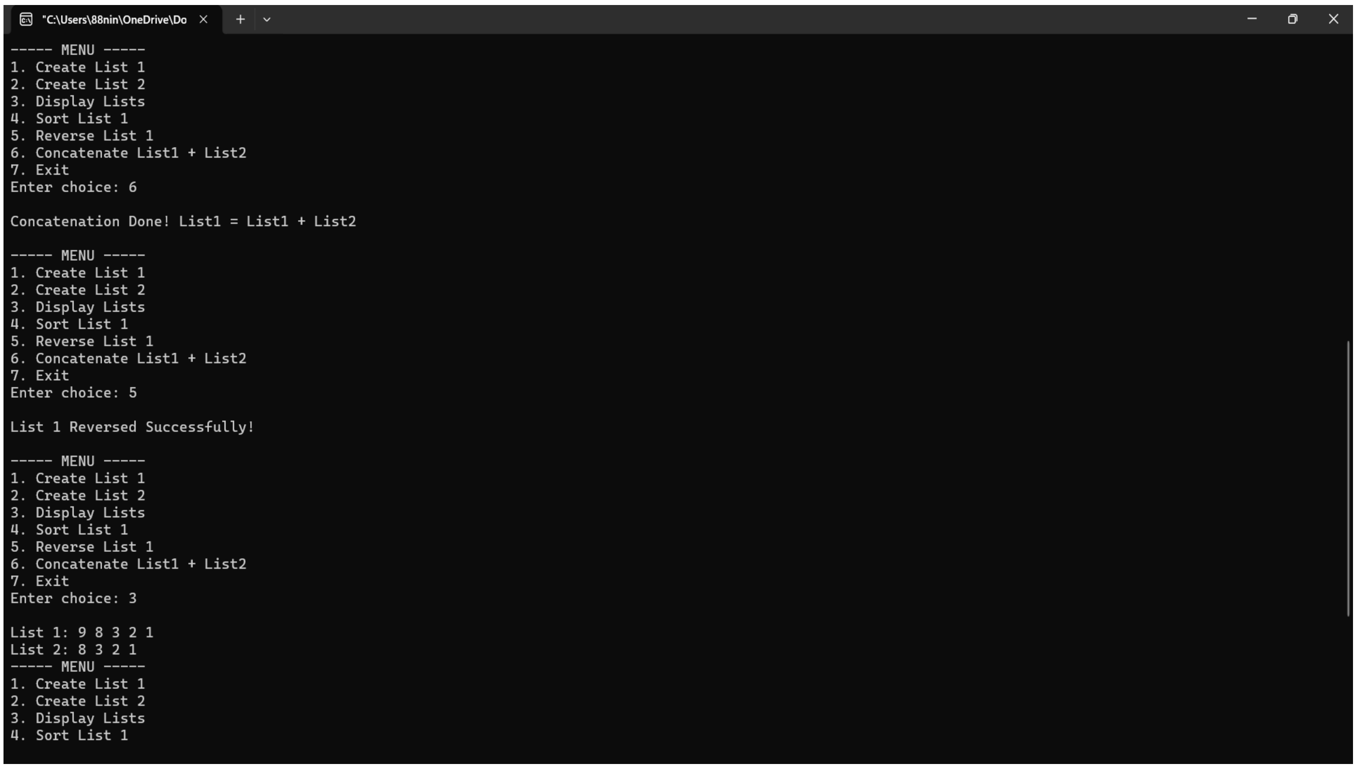
```
        case 4:
          list1 = sortList(list1);
          printf("\nList 1 Sorted Successfully!\n");
          break;

        case 5:
          list1 = reverseList(list1);
          printf("\nList 1 Reversed Successfully!\n");
          break;

        case 6:
          list1 = concatenate(list1, list2);
          printf("\nConcatenation Done! List1 = List1 + List2\n");
          break;

        case 7:
          exit(0);

        default:
          printf("\nInvalid Choice!\n");
        }
    }
    return 0;
}
```

```
}

struct node* createList() {
    struct node *start = NULL, *p, *temp;
    int item;

    printf("\nEnter elements (-999 to stop): ");

    while (1) {
        scanf("%d", &item);
        if (item == -999)
            break;

        temp = (struct node*)malloc(sizeof(struct node));
        temp->info = item;
        temp->next = NULL;

        if (start == NULL)
            start = temp;
        else {
            p = start;
            while (p->next != NULL)
                p = p->next;
            p->next = temp;
        }
    }
    return start;
}

void display(struct node *start) {
    struct node *p = start;

    if (p == NULL) {
        printf("Empty");
        return;
    }

    while (p != NULL) {
        printf("%d ", p->info);
        p = p->next;
    }
}

struct node* sortList(struct node *start) {
    struct node *i, *j;
    int temp;

    for (i = start; i != NULL; i = i->next) {
        for (j = i->next; j != NULL; j = j->next) {
            if (i->info > j->info) {
                temp = i->info;

                i->info = j->info;

                j->info = temp;

            }

        }

    }


    return start;

}


struct node* reverseList(struct node *start) {

    struct node *prev = NULL, *curr = start, *next;


    while (curr != NULL) {

        next = curr->next;

        curr->next = prev;

        prev = curr;

        curr = next;

    }


    return prev;

}


struct node* concatenate(struct node *start1, struct node *start2) {

    struct node *p;
    if (start1 == NULL) // If first list is empty
        return start2;

    p = start1;
    while (p->next != NULL)
        p = p->next;

    p->next = start2;

    return start1;
}
```

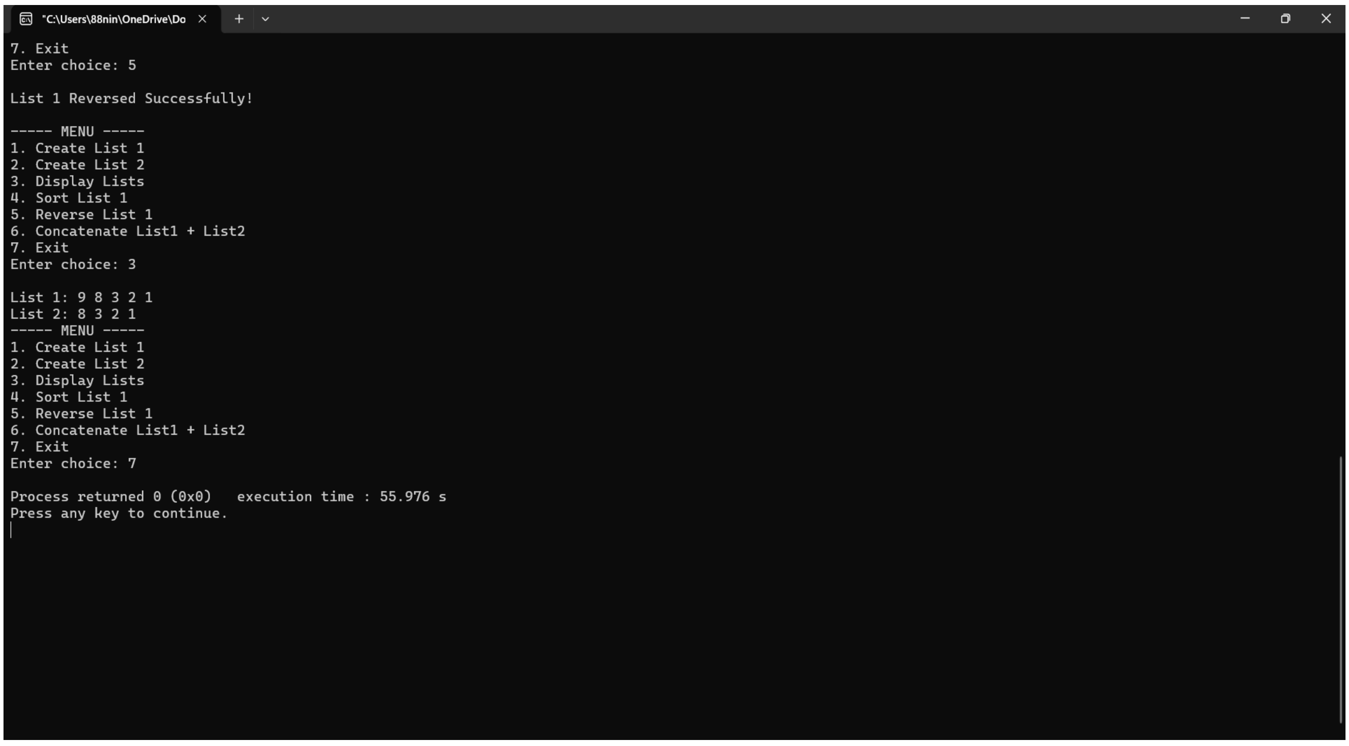
OUTPUT:-



```
----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 1

Enter elements (-999 to stop): 1
2
3
-999

----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 2

Enter elements (-999 to stop): 8
9
-999

----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 4

List 1 Sorted Successfully!
```



```
----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 6

Concatenation Done! List1 = List1 + List2

----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 5

List 1 Reversed Successfully!

----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 3

List 1: 9 8 3 2 1
List 2: 8 3 2 1
----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
```



```
7. Exit
Enter choice: 5

List 1 Reversed Successfully!

----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 3

List 1: 9 8 3 2 1
List 2: 8 3 2 1
----- MENU -----
1. Create List 1
2. Create List 2
3. Display Lists
4. Sort List 1
5. Reverse List 1
6. Concatenate List1 + List2
7. Exit
Enter choice: 7

Process returned 0 (0x0)   execution time : 55.976 s
Press any key to continue.
```