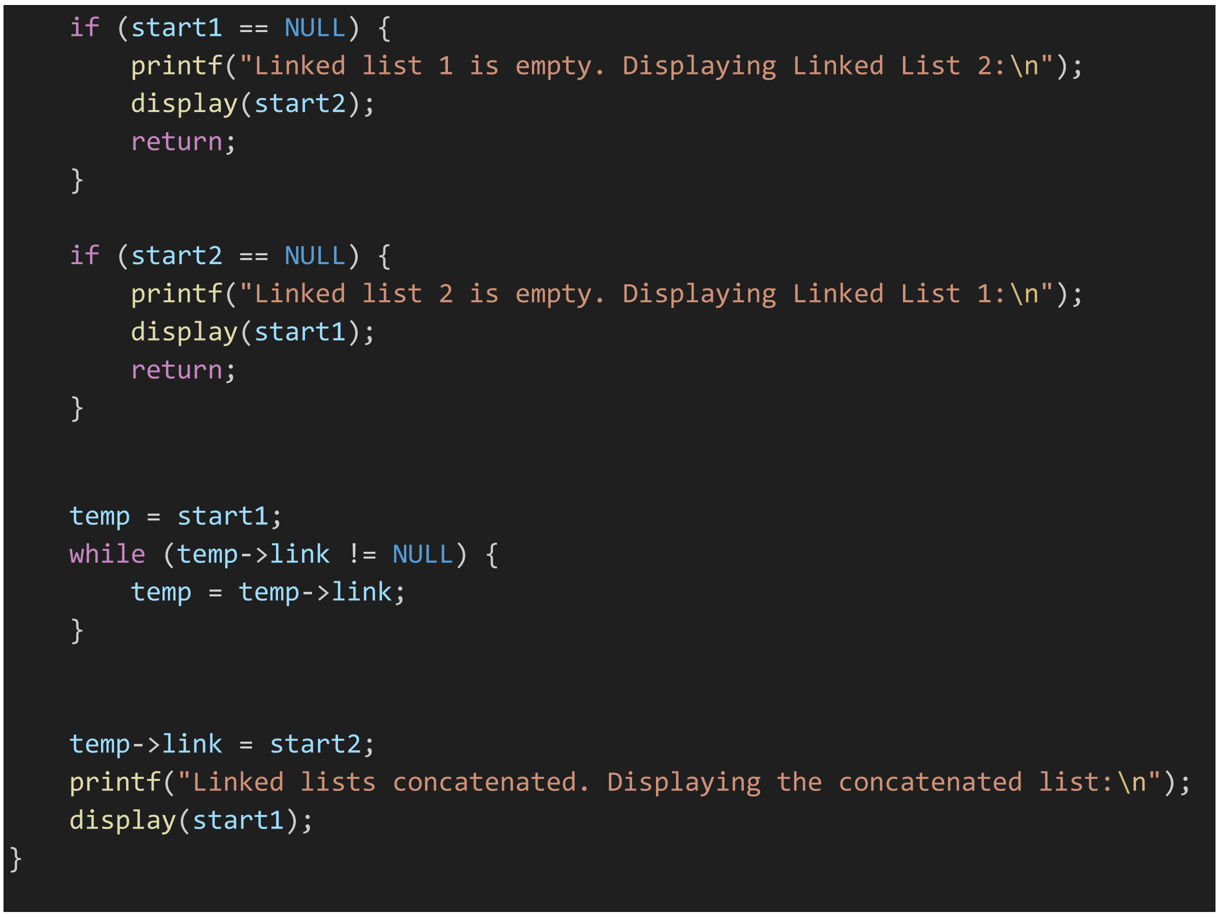
Lab 6

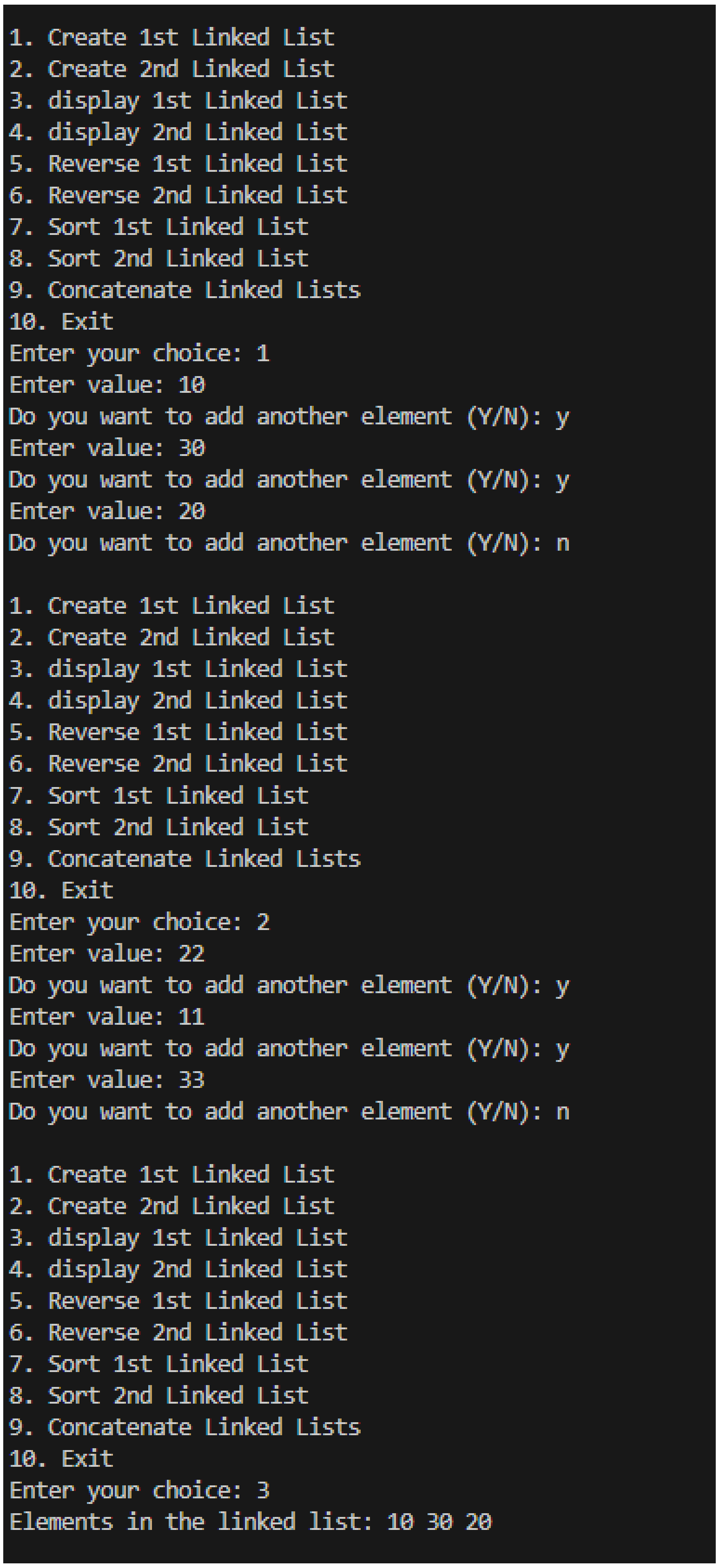
Reverse Sort and concatenate

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


struct Node {
    int data;
    struct Node *link;
};
typedef struct Node node;

node *start1 = NULL, *start2 = NULL, *newNode, *temp, *curr;


void create(node **start);
void display(node *start);
void reverse(node **start);
void sort(node *start);
void concatenate(node *start1, node *start2);

int main() {
    int choice;
    while (1) {

        printf("\n1. Create 1st Linked List \n2. Create 2nd Linked List \n3. display 1st Linked List \n4. display 2nd Linked List \n5. Reverse 1st Linked List \n6. Reverse 2nd Linked List \n7. Sort 1st Linked List \n8. Sort 2nd Linked List\n9. Concatenate Linked Lists\n10. Exit\n");
        printf("Enter your choice: ");
        if (scanf("%d", &choice) != 1) {  // Input validation for menu choice
            printf("Invalid input! Please enter a valid option.\n");
            while (getchar() != '\n');  // Clear input buffer
            continue;
        }

        switch(choice) {
            case 1: create(&start1); break;
            case 2: create(&start2); break;
            case 3: display(start1); break;
            case 4: display(start2); break;
            case 5: reverse(&start1); break;
            case 6: reverse(&start2); break;
            case 7: sort(start1); break;
            case 8: sort(start2); break;
            case 9: concatenate(start1, start2); break;
            case 10: exit(0); break;
            default: printf("Invalid choice! Try again.\n");
        }
    }
}


void create(node **start) {
    char choice;
    node *newNode;
    do {
        newNode = (node*) malloc(sizeof(node));
        if (!newNode) {
            printf("Memory allocation failed!\n");
            return;
        }

        printf("Enter value: ");
        if (scanf("%d", &newNode->data) != 1) {  // Input validation for node
data
            printf("Invalid input! Please enter a valid integer.\n");
            free(newNode);  // Free the allocated memory on invalid input
            while (getchar() != '\n');  // Clear input buffer
            continue;
        }
        newNode->link = NULL;

        if (*start == NULL) {
            *start = newNode;
            curr = newNode;
        } else {
            curr->link = newNode;
            curr = newNode;
        }

        printf("Do you want to add another element (Y/N): ");
        scanf(" %c", &choice);
    } while(choice == 'y' || choice == 'Y');
}


void display(node *start) {
    if (start == NULL) {
        printf("Linked list is empty.\n");
        return;
    }

    temp = start;
    printf("Elements in the linked list: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->link;
    }
    printf("\n");
}


void reverse(node **start) {
    node *prev = NULL, *current = *start, *next = NULL;
    while (current != NULL) {
        next = current->link;
        current->link = prev;
        prev = current;
        current = next;
    }
    *start = prev;
    printf("Linked list reversed.\n");
}


void sort(node *start) {
    if (start == NULL) {
        printf("Linked list is empty.\n");
        return;
    }

    node *i, *j;
    int tempData;
    for (i = start; i != NULL; i = i->link) {
        for (j = i->link; j != NULL; j = j->link) {
            if (i->data > j->data) {
                // Swap data
                tempData = i->data;
                i->data = j->data;
                j->data = tempData;
            }
        }
    }
    printf("Linked list sorted.\n");
}


void concatenate(node *start1, node *start2) {
    if (start1 == NULL && start2 == NULL) {
        printf("Both linked lists are empty.\n");
        return;
    }
```

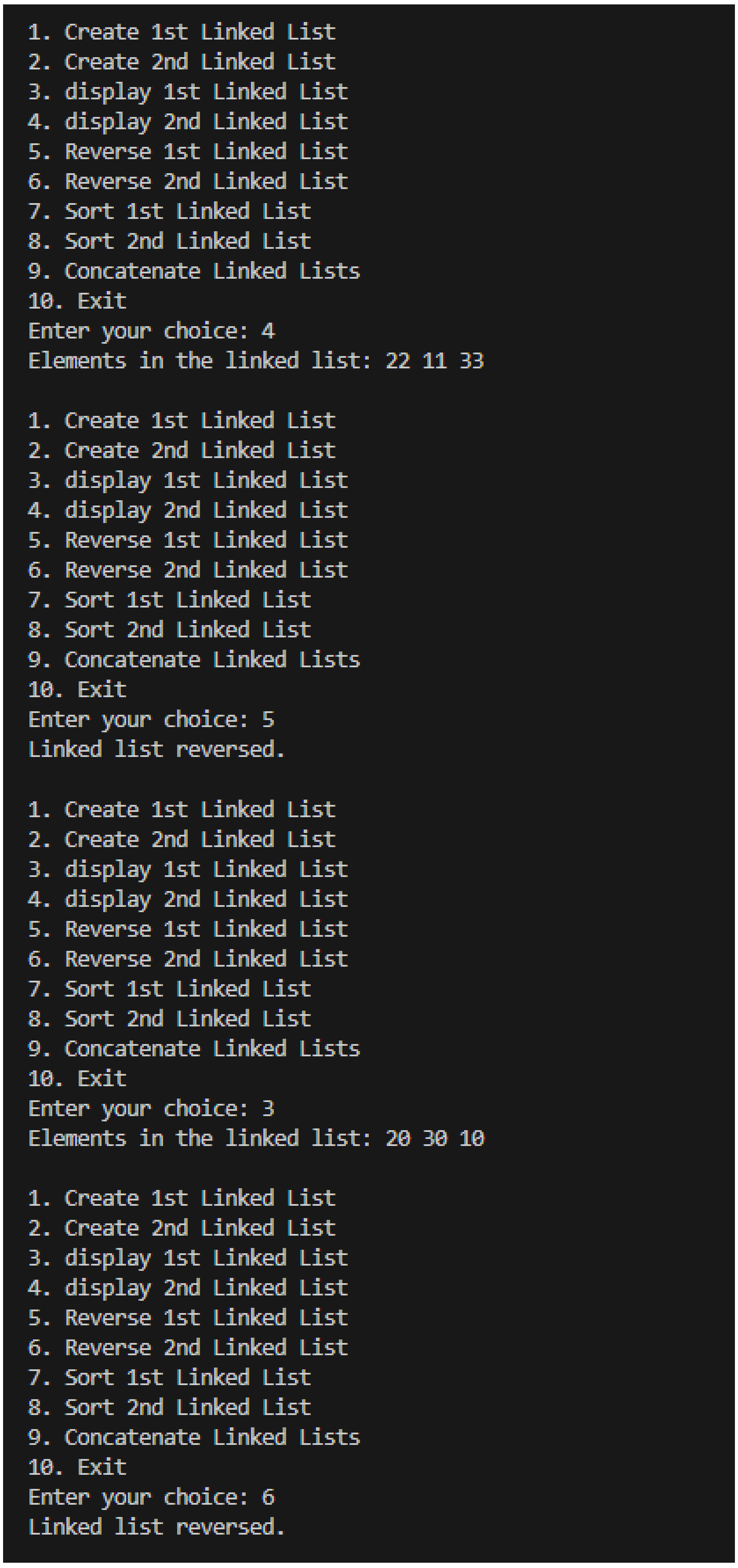


```c
    if (start1 == NULL) {
        printf("Linked list 1 is empty. Displaying Linked List 2:\n");
        display(start2);
        return;
    }

    if (start2 == NULL) {
        printf("Linked list 2 is empty. Displaying Linked List 1:\n");
        display(start1);
        return;
    }


    temp = start1;
    while (temp->link != NULL) {
        temp = temp->link;
    }


    temp->link = start2;
    printf("Linked lists concatenated. Displaying the concatenated list:\n");
    display(start1);
}
```

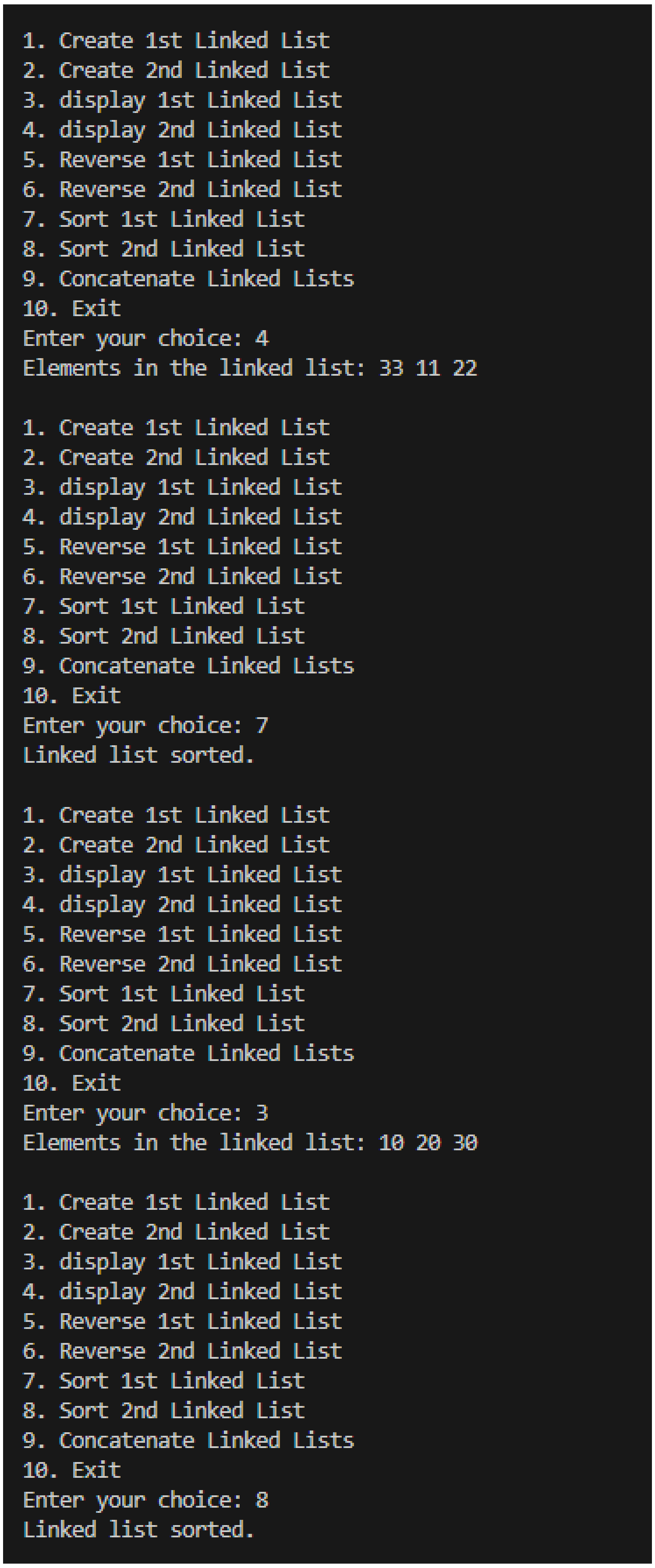
Output:



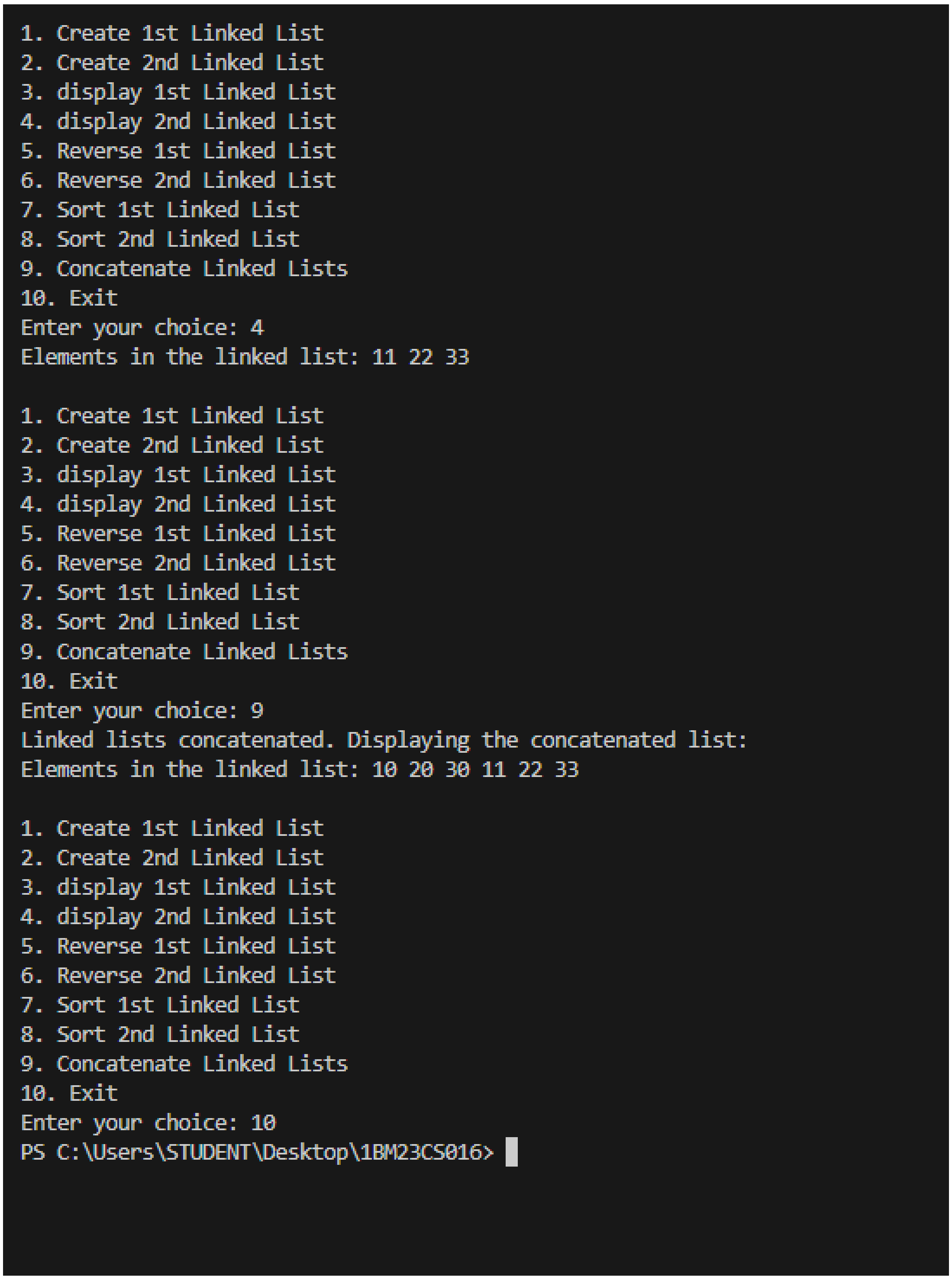
```
1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 1
Enter value: 10
Do you want to add another element (Y/N): y
Enter value: 30
Do you want to add another element (Y/N): y
Enter value: 20
Do you want to add another element (Y/N): n

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 2
Enter value: 22
Do you want to add another element (Y/N): y
Enter value: 11
Do you want to add another element (Y/N): y
Enter value: 33
Do you want to add another element (Y/N): n

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 3
Elements in the linked list: 10 30 20
```



```
1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 4
Elements in the linked list: 22 11 33

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 5
Linked list reversed.

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 3
Elements in the linked list: 20 30 10

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 6
Linked list reversed.
```



```
1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 4
Elements in the linked list: 33 11 22

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 7
Linked list sorted.

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 3
Elements in the linked list: 10 20 30

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 8
Linked list sorted.
```



```
1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 4
Elements in the linked list: 11 22 33

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 9
Linked lists concatenated. Displaying the concatenated list:
Elements in the linked list: 10 20 30 11 22 33

1. Create 1st Linked List
2. Create 2nd Linked List
3. display 1st Linked List
4. display 2nd Linked List
5. Reverse 1st Linked List
6. Reverse 2nd Linked List
7. Sort 1st Linked List
8. Sort 2nd Linked List
9. Concatenate Linked Lists
10. Exit
Enter your choice: 10
PS C:\Users\STUDENT\Desktop\1BM23CS016>
```