6a. a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

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

// Definition of node
struct node {
   int data;
   struct node *next;
};

// Function to create a linked list
struct node* create() {
   int n, value;
   struct node *head = NULL, *temp = NULL, *newnode;

   printf("Enter number of nodes: ");
   scanf("%d", &n);

   for (int i = 0; i < n; i++) {
      newnode = (struct node *)malloc(sizeof(struct node));
      printf("Enter data for node %d: ", i + 1);
      scanf("%d", &value);

      newnode->data = value;
      newnode->next = NULL;
```

```
        if (head == NULL) {
            head = temp = newnode;
        } else {
            temp->next = newnode;
            temp = newnode;
        }
    }
    return head;
}

// Display linked list
void display(struct node *head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

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

// Sort linked list (Bubble sort)
void sort(struct node *head) {
    struct node *i, *j;
```

```
    int temp;

    if (head == NULL)
        return;

    for (i = head; i->next != 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;
            }
        }
    }
    printf("Linked list sorted.\n");
}

// Reverse linked list
struct node* 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;
    }
    printf("Linked list reversed.\n");
```

```
    return prev;
}

// Concatenate two linked lists
struct node* concatenate(struct node *head1, struct node *head2) {
    struct node *temp = head1;

    if (head1 == NULL)
        return head2;

    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = head2;

    printf("Linked lists concatenated.\n");
    return head1;
}

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

    do {
        printf("\n--- Singly Linked List Operations ---\n");
        printf("1. Create List 1\n");
        printf("2. Create List 2\n");
        printf("3. Display List 1\n");
```

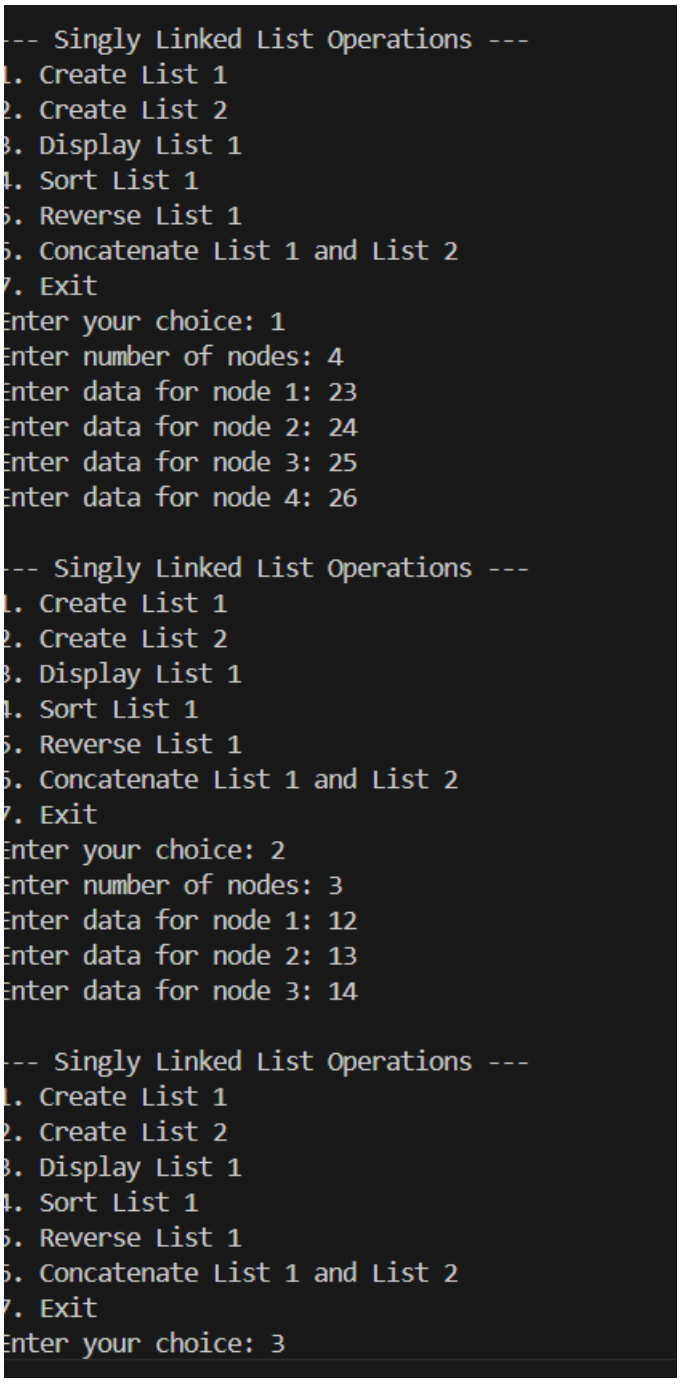
```
printf("4. Sort List 1\n");
printf("5. Reverse List 1\n");
printf("6. Concatenate List 1 and List 2\n");
printf("7. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
    case 1:
        list1 = create();
        break;
    case 2:
        list2 = create();
        break;
    case 3:
        display(list1);
        break;
    case 4:
        sort(list1);
        break;
    case 5:
        list1 = reverse(list1);
        break;
    case 6:
        list1 = concatenate(list1, list2);
        break;
    case 7:
        printf("Exiting program.\n");
```

```
            break;
        default:
            printf("Invalid choice! Try again.\n");
    }
  } while (choice != 7);

  return 0;
}
```

OUTPUT:


--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
5. Reverse List 1
6. Concatenate List 1 and List 2
7. Exit
Enter your choice: 1
Enter number of nodes: 4
Enter data for node 1: 23
Enter data for node 2: 24
Enter data for node 3: 25
Enter data for node 4: 26
--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
5. Reverse List 1
6. Concatenate List 1 and List 2
7. Exit
Enter your choice: 2
Enter number of nodes: 3
Enter data for node 1: 12
Enter data for node 2: 13
Enter data for node 3: 14
--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
5. Reverse List 1
6. Concatenate List 1 and List 2
7. Exit
Enter your choice: 3

```
Enter your choice: 3
23 -> 24 -> 25 -> 26 -> NULL

--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
5. Reverse List 1
6. Concatenate List 1 and List 2
7. Exit
Enter your choice: 4
Linked list sorted.

--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
5. Reverse List 1
6. Concatenate List 1 and List 2
7. Exit
Enter your choice: 5
Linked list reversed.

--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
5. Reverse List 1
6. Concatenate List 1 and List 2
7. Exit
Enter your choice: 6
Linked lists concatenated.

--- Singly Linked List Operations ---
1. Create List 1
2. Create List 2
3. Display List 1
4. Sort List 1
```