S.No: 1	Exp. Name: ***Operations on a Single Linked List***	**Date:2025-08-08**

Aim:

Write a menu-driven C program that implements a singly linked list for the following operations:

1. Insertion at end.
2. Display list.
3. Sort the linked list.
4. Search for an element.

Note: The driver code for taking inputs is provided in the "SLLOps2Main.c", you need to complete the functions in the file "SLLOps2.c".

Source Code:

SLLOps2Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "SLLOps2.c"

// Function prototypes
void insertAtEnd(struct Node** head, int data);
void display(struct Node* head);
void sortList(struct Node** head);
struct Node* search(struct Node* head, int key);

int main() {
    struct Node* head = NULL;
    int choice, data, key;
    struct Node* foundNode;

    printf("1. Insert\n");
    printf("2. Display\n");
    printf("3. Sort\n");
    printf("4. Search\n");
    printf("5. Exit\n");

    do {
        printf("Choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                printf("Data: ");
                scanf("%d", &data);
                insertAtEnd(&head, data);
                break;
            case 2:
                display(head);
                break;
            case 3:
                sortList(&head);
                break;
            case 4:
```

```c
                printf("Key to search: ");
                scanf("%d", &key);
                foundNode = search(head, key);
                if (foundNode != NULL)
                    printf("%d found\n", foundNode->data);
                else
                    printf("%d not found\n", key);
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 5);

    return 0;
}
```

SLLOps2.c

```c
// Node structure
struct Node {
    int data;
    struct Node* next;
};

// Function to insert a node at the end of the list
void insertAtEnd(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   if(*head==NULL){
      *head = newNode;
      return;
   }
   struct Node* temp = *head;
   while(temp->next!=NULL){
      temp = temp->next;
   }
   temp->next = newNode;
}

// Function to display the linked list
void display(struct Node* head) {
   if(head == NULL){
      printf("List is empty\n");
      return;
   }
   struct Node* temp = head;
   while(temp!=NULL){
      printf("%d -> ", temp->data);
      temp = temp->next;
   }
   printf("NULL\n");

}

// Function to sort the linked list in ascending order
void sortList(struct Node** head) {
   if(*head == NULL){
      printf("List is empty\n");
      return;
   }
   struct Node *i, *j;
   int temp;
   for(i = *head; i!=NULL && 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("List sorted successfully\n");
}

// Function to search for a node with a given key
struct Node* search(struct Node* head, int key) {
   int p = 1;
   while(head!=NULL){
      if(head->data==key){
         return head;
      }
      head = head->next;
      p++;
   }
   return NULL;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1. Insert 2
2. Display 2
3. Sort 2
4. Search 2
5. Exit 2
Choice: 2
List is empty 3
Choice: 3
List is empty 4
Choice: 4
Key to search: 0

```
0 not found 53
Choice:  53
Invalid choice 1
Choice:  1
Data:  1
Choice:  1
Data:  2
Choice:  1
Data:  3
Choice:  2
1 -> 2 -> 3 -> NULL 5
Choice:  5
Exiting...
```

Test Case - 2

User Output

```
1. Insert 1
2. Display 1
3. Sort 1
4. Search 1
5. Exit 1
Choice:  1
Data:  6
Choice:  1
Data:  5
Choice:  1
Data:  8
Choice:  1
Data:  2
Choice:  1
Data:  10
Choice:  2
6 -> 5 -> 8 -> 2 -> 10 -> NULL 3
Choice:  3
List sorted successfully 2
Choice:  2
2 -> 5 -> 6 -> 8 -> 10 -> NULL 4
Choice:  4
Key to search:  5
5 found 4
Choice:  4
Key to search:  12
12 not found 6
Choice:  6
Invalid choice 5
Choice:  5
Exiting...
```