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

struct node {
    int data;
    struct node *next;
};
struct node *head1 = NULL, *head2 = NULL;

void createList(struct node **head) {
    struct node *newNode, *temp = NULL;
    int n, data, i;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
    if (n <= 0) {
        *head = NULL;
        return;
    }
    *head = NULL;
    for (i = 1; i <= n; i++) {
        newNode = (struct node*)malloc(sizeof(struct node));
        printf("Enter data for node %d: ", i);
        scanf("%d", &data);
        newNode->data = data;
        newNode->next = NULL;
        if (*head == NULL)
            *head = newNode;
        else
            temp->next = newNode;
        temp = newNode;
    }
    printf("List created: ");
    temp = *head;
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

void displayList(struct node *head) {
    struct node *temp = head;
    if (!head) {
        printf("List is empty\n");
        return;
    }
    while (temp) {
        printf("%d -> ", temp->data);
```

```c
void displayList(struct node *head) {
    struct node *temp = head;
    if (!head) {
        printf("List is empty\n");
        return;
    }
    while (temp) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

void reverseList(struct node **head) {
    struct node *prev = NULL, *next = NULL, *cur = *head;
    while (cur) {
        next = cur->next;
        cur->next = prev;
        prev = cur;
        cur = next;
    }
    *head = prev;

    printf("Reversed list: ");
    displayList(*head);
}

void sortList(struct node **head) {
    struct node *i, *j;
    int t;
    if (!*head) {
        printf("List is empty\n");
        return;
    }
    for (i = *head; i; i = i->next)
        for (j = i->next; j; j = j->next)
            if (i->data > j->data) {
                t = i->data;
                i->data = j->data;
                j->data = t;
            }
    printf("Sorted list: ");
    displayList(*head);
}
```

```c
void concatenateList() {
    struct node *temp;
    if (!head1) {
        head1 = head2;
        return;
    }
    if (!head2) return;
    temp = head1;
    while (temp->next)
        temp = temp->next;
    temp->next = head2;
    printf("Concatenated List1 + List2: ");
    displayList(head1);
}

int main() {
    int choice, listChoice;
    while (1) {
        printf("\n1.Create List1\n2.Create List2\n3.Reverse\n4.Sort\n5.Display\n6.Concatenate\n7.Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            createList(&head1);
            break;
        case 2:
            createList(&head2);
            break;
        case 3:
            printf("Reverse which list (1/2): ");
            scanf("%d", &listChoice);
            if (listChoice == 1) reverseList(&head1);
            else reverseList(&head2);
           break;
        case 4:
            printf("Sort which list (1/2): ");
            scanf("%d", &listChoice);
            if (listChoice == 1) sortList(&head1);
            else sortList(&head2);
            break;
        case 5:
            printf("Display which list (1/2): ");
            scanf("%d", &listChoice);
            if (listChoice == 1) displayList(head1);
            else displayList(head2);
            break;
        case 6:
```

```
1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 1
Enter number of nodes: 3
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30
List created: 10 -> 20 -> 30 -> NULL

1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 2
Enter number of nodes: 3
Enter data for node 1: 11
Enter data for node 2: 22
Enter data for node 3: 33
List created: 11 -> 22 -> 33 -> NULL

1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 3
Reverse which list (1/2): 1
Reversed list: 30 -> 20 -> 10 -> NULL

1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 3
Reverse which list (1/2): 2
Reversed list: 33 -> 22 -> 11 -> NULL

1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 6
Concatenated List1 + List2: 30 -> 20 -> 10 -> 33 -> 22 -> 11 -> NULL
```

```
1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 4
Sort which list (1/2): 1
Sorted list: 10 -> 11 -> 20 -> 22 -> 30 -> 33 -> NULL

1.Create List1
2.Create List2
3.Reverse
4.Sort
5.Display
6.Concatenate
7.Exit
Enter choice: 7

Process returned 0 (0x0)   execution time : 95.776 s
Press any key to continue.
```

8) WAP to implement single linked list with following operations. sort linked list, reverse linked list, concatenation of linked list.

a) pseudocode.

// Sort the linked list

Function Sort(struct node* head) :

if (!head) nothing to sort ; return;

struct node *j, *i;

for (i = head ; i != NULL; i = i->next)

for (j = head->next ; j != NULL ; j = j->next)

if (i->data > j->data)

int temp = i->data;

i->data = j->data;

j->data = temp;

End if

End for

End for.

display sorted list head

End function

(b) reverse the linked list

Function reverse (struct node* head).

struct node *prev = NULL , *curr = head, *Next = NULL;

while (curr != NULL) {

Next = curr->next;

curr->next = prev;

prev = curr;

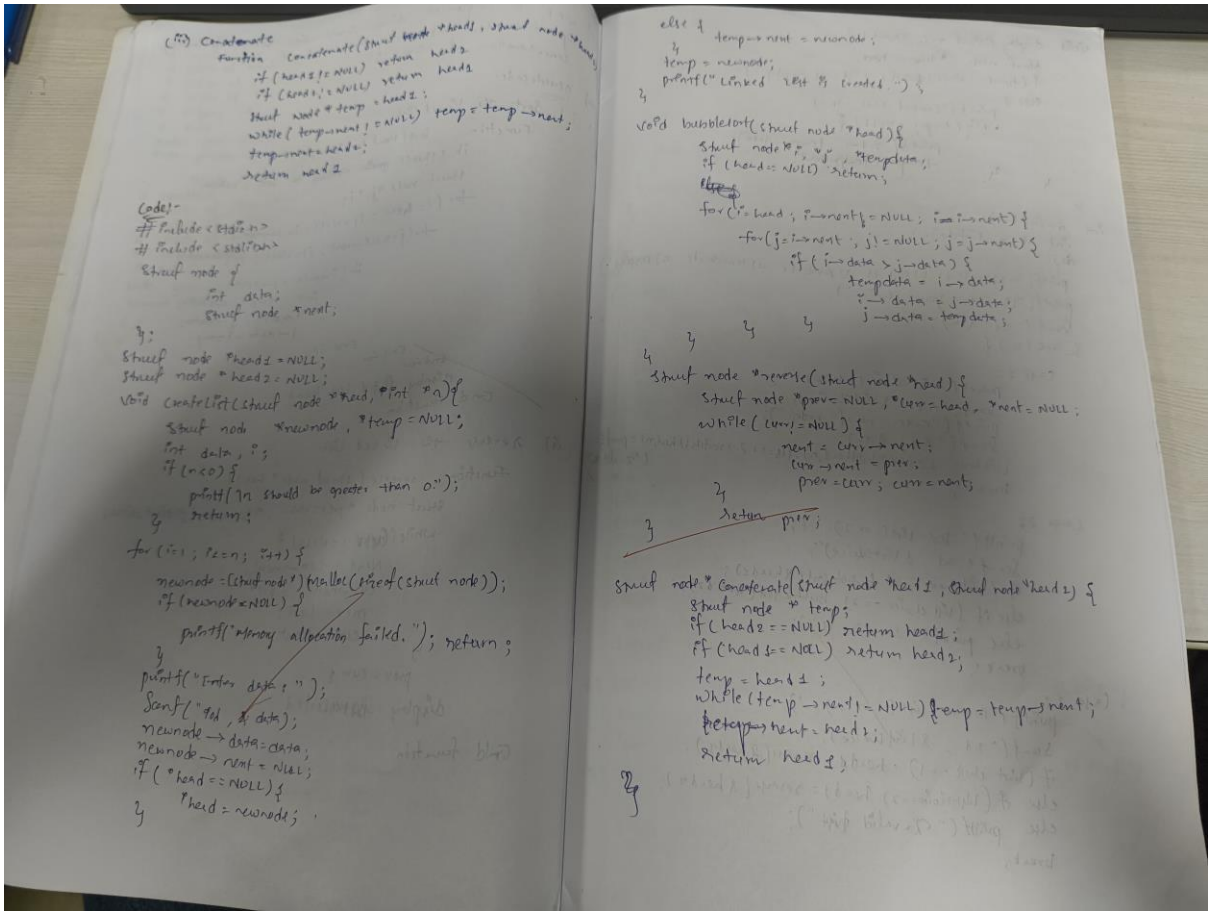
curr = Next;

}

prev = curr;

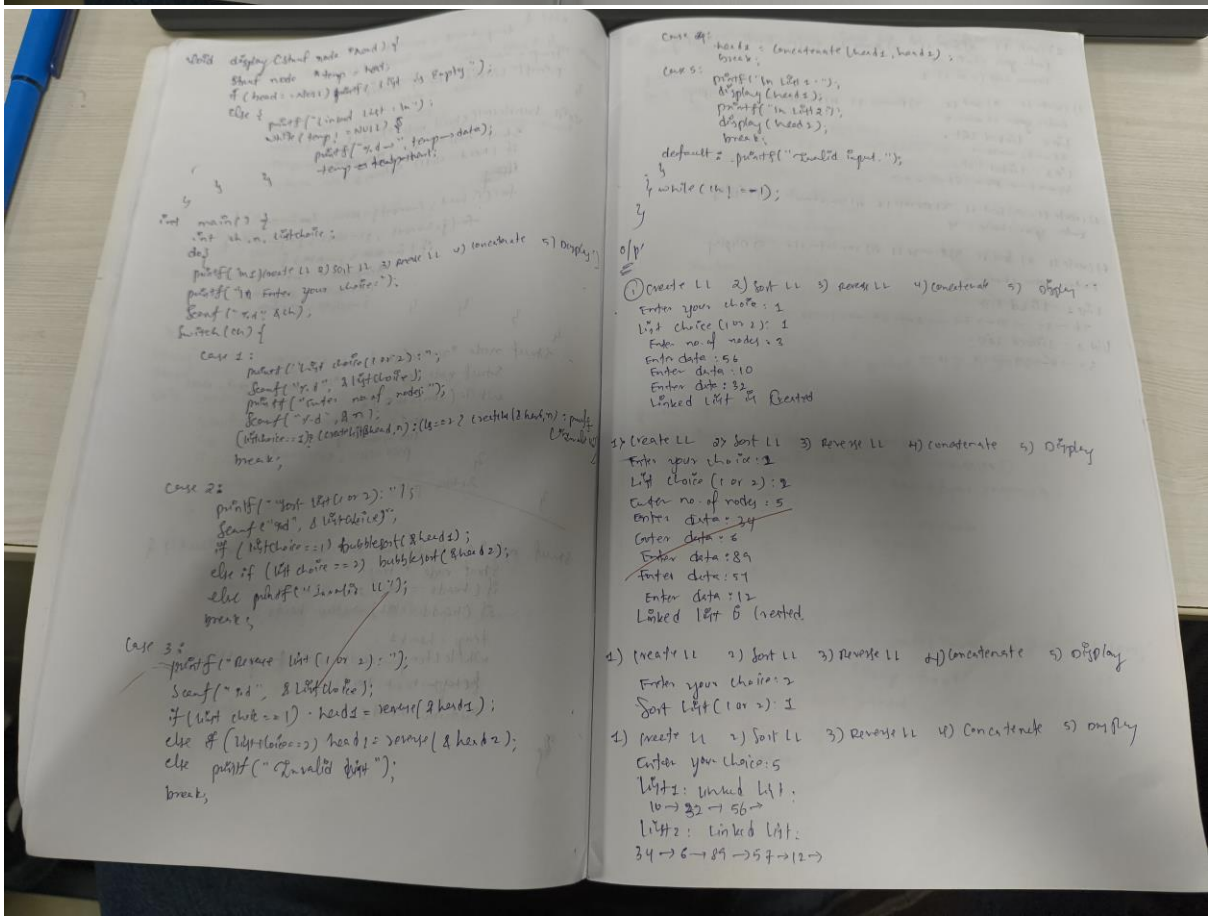
display sorted

End function



```
(iii) Concatenate
Function concatenate(struct node *head1, struct node *head2)
if (head1 != NULL) return head2
if (head2 != NULL) return head1
struct node *temp = head1;
while (temp->next != NULL) temp = temp->next;
temp->next = head2;
return head1

Code:-
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
};
struct node *head1 = NULL;
struct node *head2 = NULL;
void createList(struct node **head, int n){
    struct node *newnode, *temp = NULL;
    int data, i;
    if (n<0) {
        printf("n should be greater than 0");
        return;
    }
for (i=1; i<=n; i++) {
    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode==NULL) {
        printf("Memory allocation failed."); return;
    }
printf("Enter data: ");
scanf("%d", &data);
newnode->data = data;
newnode->next = NULL;
if (*head == NULL) {
    *head = newnode;
}
else {
    temp->next = newnode;
}
temp = newnode;
printf("Linked list is created.");
}
void bubblesort(struct node *head){
    struct node *i, *j, *tempdata;
    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) {
                tempdata = i->data;
                i->data = j->data;
                j->data = tempdata;
            }
        }
    }
struct node *reverse(struct node *head) {
    struct node *prev=NULL, *curr=head, *next=NULL;
    while (curr!=NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr; curr = next;
    }
    return prev;
}
struct node *concatenate(struct node *head1, struct node *head2) {
    struct node *temp;
    if (head2==NULL) return head1;
    if (head1==NULL) return head2;
    temp = head1;
    while (temp->next!=NULL) temp = temp->next;
    temp->next = head2;
    return head1;
}
```



```
void display(struct node *head) {
    struct node *temp = head;
    if (head==NULL) printf("List is empty");
    else {
        printf("Linked List: \n");
        while(temp!=NULL) {
            printf("%d->", temp->data);
            temp = temp->next;
        }
    }
}
int main() {
    int ch, n, ListChoice;
    do{
    printf("\n1)Create LL 2)Sort LL 3) Reverse LL 4) Concatenate 5) Display");
    printf("\n Enter your choice:");
    scanf("%d", &ch);
    switch(ch) {
        case 1:
            printf("List choice(1 or 2):");
            scanf("%d", &ListChoice);
            printf("Enter no. of nodes:");
            scanf("%d", &n);
            (ListChoice==1)? (createLL(&head1,n)):(ListChoice==2)? (createLL(&head2,n)):printf("Invalid
            break;
        case 2:
            printf("Sort List(1 or 2):");
            scanf("%d", &ListChoice);
            if (ListChoice==1) bubblesort(&head1);
            else if (ListChoice==2) bubblesort(&head2);
            else printf("Invalid LL");
            break;
Case 3:
        printf("Reverse List(1 or 2):");
        scanf("%d", &ListChoice);
        if (List choice==1) head1 = reverse(&head1);
        else if (ListChoice==2) head1 = reverse(&head2);
        else printf("Invalid input");
        break;
        case 4:
            head1 = concatenate(head1, head2);
            break;
        case 5:
            printf("In List1:");
            display(head1);
            printf("In List2:");
            display(head2);
            break;
        default: printf("Invalid input.");
    }
} while(ch!=-1);
}
```

O/P

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 1
List choice (1 or 2): 1
Enter no. of nodes: 2
Enter data: 56
Enter data: 10
Enter data: 32
Linked List is Created

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 1
List choice (1 or 2): 2
Enter no. of nodes: 5
Enter data: 34
Enter data: 6
Enter data: 89
Enter data: 57
Enter data: 12
Linked list is created

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 2
Sort List (1 or 2): 1

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 5
List1: Linked List:
10→32→56→
List2: Linked List:
34→6→89→57→12→

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 3
Reverse List (1 or 2): 1

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 5
List 1 : Linked List:
56 → 32 → 10 →
List 2 : Linked List:
34 → 6 → 89 → 57 → 12 →

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display.
Enter your choice: 4

1) Create LL 2) Sort LL 3) Reverse LL 4) Concatenate 5) Display
Enter your choice: 5
List 1 : Linked List:
56 → 32 → 10 → 34 → 6 → 89 → 57 → 12 →
List 2 : Linked List:
34 → 6 → 89 → 57 → 12 →