

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

struct Node {
    int data;
    struct Node *next;
};

struct Node *head = NULL;

void createList(int n) {
    struct Node *newNode, *temp;
    int data, i;

    if (n <= 0) {
        printf("Number of nodes should be greater than 0.\n");
        return;
    }

    for (i = 1; i <= n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        if (newNode == NULL) {
            printf("Memory allocation failed.\n");
            return;
        }

        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            temp = newNode;
        } else {
            temp->next = newNode;
            temp = newNode;
        }
    }

    printf("Unlinked List created successfully!\n");
}

void display() {
    struct Node *temp = head;

    if (temp == NULL) {
        printf("List is empty.\n");
        return;
    }

    printf("Linked List:\n");
    while (temp != NULL) {
        printf("%d-", temp->data);
        temp = temp->next;
    }
    printf("\nNULL\n");
}

void sortList() {
    struct Node *i, *j;
    int temp;

    if (head == NULL) {
        printf("List is empty.\n");
        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");
}

void reverseList() {
    struct Node *prev = NULL, *current = head, *next = NULL;
    int temp;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }

    head = prev;
}

struct Node *concatenate(struct Node *h1, struct Node *h2) {
    if (h1 == NULL) return h2;
    if (h2 == NULL) return h1;

    struct Node *temp = h1;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = h2;
}
```

```
Windows (CR+LF) WINDOWS-1252 Line 30, Col 34 Pos 816 Insert Modified Read/Write default
D:\ethan\DP\Sort_Reverse_Concat.c 08:12 08-12-2023
```

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

struct Node {
    int data;
    struct Node *next;
};

struct Node *head = NULL;

void createList(int n) {
    struct Node *newNode, *temp;
    int data, i;

    if (n <= 0) {
        printf("Number of nodes should be greater than 0.\n");
        return;
    }

    for (i = 1; i <= n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        if (newNode == NULL) {
            printf("Memory allocation failed.\n");
            return;
        }

        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            temp = newNode;
        } else {
            temp->next = newNode;
            temp = newNode;
        }
    }

    printf("Unlinked List created successfully!\n");
}

void display() {
    struct Node *temp = head;

    if (temp == NULL) {
        printf("List is empty.\n");
        return;
    }

    printf("Linked List:\n");
    while (temp != NULL) {
        printf("%d-", temp->data);
        temp = temp->next;
    }
    printf("\nNULL\n");
}

void sortList() {
    struct Node *i, *j;
    int temp;

    if (head == NULL) {
        printf("List is empty.\n");
        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");
}

void reverseList() {
    struct Node *prev = NULL, *current = head, *next = NULL;
    int temp;

    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }

    head = prev;
}

struct Node *concatenate(struct Node *h1, struct Node *h2) {
    if (h1 == NULL) return h2;
    if (h2 == NULL) return h1;

    struct Node *temp = h1;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = h2;
}
```

Windows (CR+LF) WINDOWS-1252 Line 30, Col 34 Pos 816 Insert Modified Read/Write default
D:\ethan\DP\Sort_Reverse_Concat.c 08:13 08-12-2023

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

struct Node {
    int data;
    struct Node *next;
};

void printList(struct Node *node) {
    while (node != NULL) {
        printf("%d\n", node->data);
        node = node->next;
    }
}

int main() {
    struct Node *head = NULL;
    int choice;
    int n, m;
    int i;

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

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(&head, n);
                break;

            case 2:
                display();
                break;

            case 3:
                sortList();
                break;

            case 4:
                reverseList();
                break;

            case 5:
                struct Node *head2 = NULL;
                int m;
                printf("Enter number of nodes for second list: ");
                scanf("%d", &m);

                struct Node *tempHead = head;
                head = NULL;
                createList(&head, m);

                while (tempHead != NULL) {
                    struct Node *tempNode = tempHead;
                    tempHead = tempHead->next;
                    tempNode->next = head;
                    head = tempNode;
                }

                printList(head);
                break;
        }
    }
}
```

D:\NethanDIP\Sort_Reverse_Concat.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 30, Col 34 Pos 816 Insert Modified Read/Write default ENG IN 08:13 08-12-2023

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

struct Node {
    int data;
    struct Node *next;
};

void printList(struct Node *node) {
    while (node != NULL) {
        printf("%d\n", node->data);
        node = node->next;
    }
}

int main() {
    struct Node *head = NULL;
    int choice;
    int n, m;
    int i;

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

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(&head, n);
                break;

            case 2:
                display();
                break;

            case 3:
                sortList();
                break;

            case 4:
                reverseList();
                break;

            case 5:
                struct Node *head2 = NULL;
                int m;
                printf("Enter number of nodes for second list: ");
                scanf("%d", &m);

                struct Node *tempHead = head;
                head = NULL;
                createList(&head, m);

                while (tempHead != NULL) {
                    struct Node *tempNode = tempHead;
                    tempHead = tempHead->next;
                    tempNode->next = head;
                    head = tempNode;
                }

                printList(head);
                break;
        }
    }
}
```

D:\NethanDIP\Sort_Reverse_Concat.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 30, Col 34 Pos 816 Insert Modified Read/Write default ENG IN 08:13 08-12-2023

```
C:\Users\User\Downloads\Sort_Reverse_Concat.exe
-----
Enter your choice: 1
Enter number of nodes: 4
Enter data for node 1: 32
Enter data for node 2: 23
Enter data for node 3: 56
Enter data for node 4: 78
Linked List created successfully!

----- MENU -----
1. Create Linked List
2. Display List
3. Sort List
4. Reverse List
5. Concatenate Two Lists
6. Exit
-----
Enter your choice: 2
Linked List: 32->23->56->78->NULL

----- MENU -----
1. Create Linked List
2. Display List
3. Sort List
4. Reverse List
5. Concatenate Two Lists
6. Exit
-----
Enter your choice: 3
Linked List sorted.

----- MENU -----
1. Create Linked List
2. Display List
3. Sort List
4. Reverse List
5. Concatenate Two Lists
6. Exit
-----
Enter your choice: 4
Linked List reversed.

----- MENU -----
1. Create Linked List
2. Display List
3. Sort List
4. Reverse List
5. Concatenate Two Lists
6. Exit
-----
Enter your choice:
```

Windows Type here to search 21°C Partly cloudy 20:45
2025 ENG 08-12-2025

b) WAP to implement single linked list with following operation. Sort linked list, reverse linked list, concatenation of linked list

Pseudocode:-

Function sort

if list.head == null or list.head.next == null
return

swapped = true

while swapped == true:

 swapped = false

 current = list.head

Function sort

sort (struct node * head):

if (head == null): print nothing to sort

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;

 endif

function reverse

reverse (struct node * head)

struct node * previous, * current; 'n

while (curr != null):

 next = curr->next;

 curr->next = prev;

 prev = curr;

}

 curr = next;

DATE:

prev curr
display reversed
end

Concatenation
Function concatenation (struct node* heads; struct node*
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* head = null;

```
void createlist(int n) {
    struct node* newnode, * temp;
    int data, i;

    if (n <= 0) {
        printf("Number of nodes should greater than 0");
        return;
    }
}
```

DATE: PAGE:

```
for (i=1; i<=n; i++) {
    newnode = (struct node*) malloc (sizeof (struct node));
    if (newnode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    printf("Enter data for node %d: ", i);
    scanf("%d", &data);
    newnode->data = data;
    newnode->next = NULL;
    if (head == NULL) {
        head = newnode;
        temp = newnode;
    } else {
        temp->next = newnode;
        temp = newnode;
    }
}
```

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

```

void sortlist() {
    struct Node *i, *j;
    int temp;
    if (head == NULL) {
        printf("list is empty.\n");
        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");
}

```

```

void revercelist() {
    struct Node *prev = NULL, *current = head, *next = NULL;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
    printf("linked list reversed.\n");
}

```

struct node * concatenate(struct node * h1, struct node * h2){
if (h2 == NULL) return h2;
if (h1 == NULL) return h1;
struct node * temp = h1;
while (temp->next != NULL) {
temp = temp->next;
}
temp->next = h2;
return h1;

int main(){
int n, choice;
while (1){
printf("1. Create linked list\n");
printf("2. Display list\n");
printf("3. Sort list\n");
printf("4. Reverse list\n");
printf("5. Concatenate\n");
printf("6. Exit\n");
printf("Enter your choice:");
scanf("%d", &choice);
switch (choice){

CASE 1:

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

CASE 2:

display();
break;

CASE 3:

sortList();

Case 4:

reverseList();

break;

case 5:

struct Node *head = NULL;

int m;

printf("Enter number of nodes to send list:");

scanf("%d", &m);

struct node *tempHead = head;

head = NULL;

createList(m);

head2 = head;

head = tempHead;

head = concatNode(head, head2);

break;

}

Case 6:

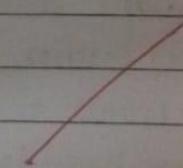
printf("Exiting\n");

exit(0);

{

return 0;

}



DATE:

Output:

Enter your choice 1

Enter number of nodes: 3

Enter data for node 1: 3

Enter data for node 2: 2

Enter data for node 3: 4

Enter your choice 2

Linked list: 3->2->4->null

Enter your choice 3

linked list sorted

Enter your choice 4

linked list reversed

Enter your choice 2

linked list: 4->3->2->null

Enter your choice 5

Enter number of nodes: 3

Enter data for node 1: 34

Enter data for node 2: 32

Enter data for node 3: 23

lists concatenated successfully.

Enter your choice 12

linked list: 4->3->2->34->32->23->null.

8/11/15
Dev