

```
"Sort_Reverse_Concat.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocuBlocks Settings Help
Start here X "Sort_Reverse_Concat.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node *next;
7 }
8
9 struct Node *head = NULL;
10
11
12 void createList(int n) {
13     struct Node *newNode, *temp;
14     int data, i;
15
16     if (n <= 0) {
17         printf("Number of nodes should be greater than 0.\n");
18         return;
19     }
20
21     for (i = 1; i <= n; i++) {
22         newNode = (struct Node*)malloc(sizeof(struct Node));
23         if (newNode == NULL) {
24             printf("Memory allocation failed.\n");
25             return;
26         }
27
28         printf("Enter data for node %d: ", i);
29         scanf("%d", &data);
30
31         newNode->data = data;
32         newNode->next = NULL;
33
34         if (head == NULL) {
35             head = newNode;
36             temp = newNode;
37         } else {
38             temp->next = newNode;
39             temp = newNode;
40         }
41     }
42
43     printf("\nLinked List created successfully.\n");
44 }
45
46 void display() {
47     struct Node *temp = head;
48
49     if (temp == NULL) {
50         printf("List is empty.\n");
51         return;
52     }
53
54     printf("Linked List: ");
55 }
```

```
"Sort_Reverse_Concat.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocuBlocks Settings Help
Start here X "Sort_Reverse_Concat.c X
52 }
53
54 printf("Linked List: ");
55 while (temp != NULL) {
56     printf("%d->", temp->data);
57     temp = temp->next;
58 }
59 printf("\n");
60
61
62 void sortList() {
63     struct Node *l, *j;
64     int temp;
65
66     if (head == NULL) {
67         printf("List is empty.\n");
68         return;
69     }
70
71     for (l = head; l->next != NULL; l = l->next) {
72         for (j = l->next; j != NULL; j = j->next) {
73             if (l->data > j->data) {
74                 temp = l->data;
75                 l->data = j->data;
76                 j->data = temp;
77             }
78         }
79     }
80
81     printf("Linked List sorted.\n");
82 }
83
84 void reverseList() {
85     struct Node *prev = NULL, *current = head, *next = NULL;
86
87     while (current != NULL) {
88         next = current->next;
89         current->next = prev;
90         prev = current;
91         current = next;
92     }
93
94     head = prev;
95
96     printf("Linked List reversed.\n");
97 }
98
99
100 struct Node* concatenate(struct Node *h1, struct Node *h2) {
101     if (h1 == NULL) return h2;
102     if (h2 == NULL) return h1;
103
104     struct Node *temp = h1;
105     while (temp->next != NULL) {
106 }
```

```
"Sort_Reverse_Concat" - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DocyBlocks Settings Help
Start here X: "Sort_Reverse_Concat" X
103
104 struct Node *temp = h1;
105 while (temp->next != NULL) {
106     temp = temp->next;
107 }
108 temp->next = h2;
109 return h1;
110 }
111
112
113 int main() {
114     int n, choice;
115
116     while (1) {
117         printf("\n----- MENU ----- \n");
118         printf("1. Create Linked List\n");
119         printf("2. Display List\n");
120         printf("3. Sort List\n");
121         printf("4. Reverse List\n");
122         printf("5. Concatenate two lists\n");
123         printf("6. Exit\n");
124         printf("Enter your choice: ");
125         scanf("%d", &choice);
126
127         switch (choice) {
128             case 1:
129                 printf("Enter number of nodes: ");
130                 scanf("%d", &n);
131                 createList(n);
132                 break;
133             case 2:
134                 display();
135                 break;
136             case 3:
137                 sortList();
138                 break;
139             case 4:
140                 reverseList();
141                 break;
142             case 5:
143                 {
144                     struct Node *head2 = NULL;
145                     int m;
146                     printf("Enter number of nodes for second list: ");
147                     scanf("%d", &m);
148
149                     struct Node *tempHead = head;
150                     head = NULL;
151                     createList(m);
152
153                     temp = tempHead;
154                     head = temp;
155                     temp = temp->next;
156                     while (temp != NULL) {
157                         temp->next = head;
158                         head = temp;
159                         temp = temp->next;
160                     }
161                     printf("List concatenated successfully.\n");
162                     break;
163                 }
164             case 6:
165                 printf("Exiting...\n");
166                 exit(0);
167             default:
168                 printf("Invalid choice! Try again.\n");
169                 continue;
170         }
171     }
172     return 0;
173 }
```

```
"Sort_Reverse_Concat" - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DocyBlocks Settings Help
Start here X: "Sort_Reverse_Concat" X
126 scanf("%d", &choice);
127
128 switch (choice) {
129     case 1:
130         printf("Enter number of nodes: ");
131         scanf("%d", &n);
132         createList(n);
133         break;
134     case 2:
135         display();
136         break;
137     case 3:
138         sortList();
139         break;
140     case 4:
141         reverseList();
142         break;
143     case 5:
144         {
145             struct Node *head2 = NULL;
146             int m;
147             printf("Enter number of nodes for second list: ");
148             scanf("%d", &m);
149
150             struct Node *tempHead = head;
151             head = NULL;
152             createList(m);
153
154             head2 = head;
155             head = tempHead;
156             temp = tempHead;
157             while (temp != NULL) {
158                 temp->next = head;
159                 head = temp;
160                 temp = temp->next;
161             }
162             printf("List concatenated successfully.\n");
163             break;
164         }
165     case 6:
166         printf("Exiting...\n");
167         exit(0);
168     default:
169         printf("Invalid choice! Try again.\n");
170         continue;
171 }
172 return 0;
173 }
```

```
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:
```

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 = j;

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

Function reverse

~~reverse~~ ( struct node \* head)

struct node \* prev = null, \* curr = head; 'n

while (curr != null):

next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

```

prev = curr;
display reversed list
end

```

### Concatenation

```

Function concatenation (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* 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");
        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: ");
while (temp != NULL) {
    printf("%d->", 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 ReverseList() {
    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 (h1 == NULL) return h2;
    if (h2 == 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 5:

reverse List()

break;

Case 5:

struct Node \*head2 = NULL;

int m;

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

scanf("%d", &amp;m);

struct Node \*tempHead = head;

head = NULL;

createList(m);

head2 = head;

head = tempHead;

head = concatenate(head, head2);

break;

}

Case 6:

printf("Exiting...");

exit(0);

}

}

return 0;

}

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: 2 → 3 → 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 of node 1: 34

Enter data of node 2: 32

Enter data for node 3: 23

lists concatenated successfully.

Enter your choice: 2

linked list: 4 → 3 → 2 → 34 → 32 → 23 → null

*Shubham*  
*Sen*