

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  struct node
5  {
6      int data;
7      struct node *next;
8  };
9
10 void insertBack(struct node **headptr,int value)
11 {
12     struct node *newnode,*temp;
13     newnode = (struct node*)malloc(sizeof(struct node));
14     newnode->data = value;
15     newnode->next = NULL;
16     temp = *headptr;
17     if(temp == NULL)
18     {
19         *headptr = newnode;
20     }
21     else
22     {
23         while(temp->next != NULL)
24             temp = temp->next;
25         temp->next = newnode;
26     }
27 }
28 void removeBack(struct node **headptr)
29 {
30     struct node *temp;
31     temp = *headptr;
32     if(temp == NULL)
33     {
34         printf("The list is Empty!!!\n");
35         return;
36     }
37     else
38     {
39         while((temp->next)->next != NULL)
40             temp = temp->next;
41         temp->next = NULL;
42         printf("Last Element has been Deleted\n");
43     }
```

```

44 }
45 void display(struct node *temp)
46 {
47     if(temp == NULL)
48     {
49         printf("The list is Empty!!!\n");
50         return;
51     }
52     else
53     {
54         while(temp!=NULL)
55         {
56             printf("%d\t",temp->data);
57             temp = temp->next;
58         }
59         printf("\n");
60     }
61 }
62 void sort(struct node **headptr)
63 {
64     struct node *p,*q;
65     p = *headptr;
66     int temp;
67     if(p == NULL)
68     {
69         printf("List is Empty!!!\n");
70         return;
71     }
72     for(; p!=NULL; p=p->next)
73     {
74         for(q=p->next;q!=NULL;q=q->next)
75         {
76             if(p->data > q->data)
77             {
78                 temp = q->data;
79                 q->data = p->data;
80                 p->data = temp;
81             }
82         }
83     }
84     printf("Sort complete!!!\n");
85 }
86 void reverse(struct node *temp)

```

```

87  {
88      if(temp == NULL)
89      {
90          printf("List is Empty!!!\n");
91          return;
92      }
93      struct node *first=NULL,*second = temp,*third=NULL;
94      while(second != NULL)
95      {
96          third = second->next;
97          second->next = first;
98          first = second;
99          second = third;
100     }
101     temp = first;
102     printf("After reversal:\n");
103     while(temp != NULL)
104     {
105         printf("%d\t",temp->data);
106         temp = temp->next;
107     }
108     printf("\n");
109 }
110 void concatenate(struct node *temp1, struct node *temp2)
111 {
112     if(temp1 == NULL && temp2 == NULL)
113     {
114         printf("Both lists are empty!!!\n");
115     }
116     else if(temp2 == NULL && temp1 != NULL)
117     {
118         printf("After concatenation:\n");
119         while(temp1 != NULL)
120         {
121             printf("%d\t",temp1->data);
122             temp1 = temp1->next;
123         }
124         printf("\n");
125     }
126     else if(temp1 == NULL && temp2 != NULL)
127     {
128         printf("After concatenation:\n");
129         while(temp2 != NULL)

```

```

130     {
131         printf("%d\t",temp2->data);
132         temp2 = temp2->next;
133     }
134     printf("\n");
135 }
136 else
137 {
138     struct node *ref = temp1;
139     while(temp1->next != NULL)
140         temp1 = temp1->next;
141     temp1->next = temp2;
142     printf("After concatenation:\n");
143     temp1 = ref;
144     while(temp1 != NULL)
145     {
146         printf("%d\t",temp1->data);
147         temp1 = temp1->next;
148     }
149     printf("\n");
150 }
151 }
152 int main(int argc,char **argv)
153 {
154     struct node *head1 = NULL, *head2 = NULL;
155     int choice,ele;
156     while(choice != 12)
157     {
158         printf("Enter choice:\n1)insert list1 2)delete list 3)display list1\n4)insert list2 5)delete list2 6)display list2\n7)sort list1 8)sort list2 9)reverse list1 10)reverse list2 11)concatenate\n12)exit\n");
159         scanf("%d",&choice);
160         switch(choice)
161         {
162             case 1:printf("Enter value:");scanf("%d",&ele);insertBack(&head1,ele);break;
163             case 2:removeBack(&head1);break;
164             case 3:display(head1);break;
165             case 4:printf("Enter value:");scanf("%d",&ele);insertBack(&head2,ele);break;
166             case 5:removeBack(&head2);break;
167             case 6:display(head2);break;
168             case 7:sort(&head1);break;
169             case 8:sort(&head2);break;
170             case 9:reverse(head1);break;
171             case 10:reverse(head2);break;
172             case 11:concatenate(head1,head2);break;

```