main.c

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

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

void create(struct node **hptr);
void display(struct node *hptr);
void reverse(struct node **hptr);
void sort(struct node *hptr);
void concatenate(struct node *hptr1, struct node *hptr2);


int main(int argc, char **argv)
{
    struct node *head1=NULL;
    struct node *head2=NULL;
    int choice,ele,choice1;
    while(choice1!=4)
    {
        printf("1. List1 \n2. List2 \n3.Concatenate \n4.Display\n");
        printf("Enter your choice:");
        scanf("%d",&choice1);

        if(choice1 == 1)
        {
            printf("List1\n");
            while(choice!=5)
```

```
34          printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display
            \n5. Quit\n");
35          printf("Enter your choice : ");
36          scanf("%d",&choice);
37          if(choice == 1)
38          {
39              create(&head1);
40          }
41          else if(choice == 2)
42          {
43              sort(head1);
44          }
45          else if(choice == 3)
46          {
47              reverse(&head1);
48          }
49          else if(choice == 4)
50              display(head1);
51          else if(choice == 5)
52              break;
53          }
54      }
55      else if(choice1 == 2)
56      {
57          int choice2;
58          printf("List 2\n");
59        while(choice2!=5)
60          {
61          printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display
            \n5. Quit");
62          printf("Enter your choice : ");
63          scanf("%d",&choice2);
```

main.c

```
64              if(choice2 == 1)
65              {
66                  create(&head2);
67              }
68              else if(choice2 == 2)
69              {
70                  sort(head2);
71              }
72              else if(choice2 == 3)
73              {
74                  reverse(&head2);
75              }
76              else if(choice2 == 4)
77                  display(head2);
78              else if(choice2 == 5)
79                  break;
80              }
81          }
82          else if(choice1 == 3)
83              concatenate(head1, head2);
84          else if(choice1 == 4)
85              display(head1);
86          else if(choice1 == 5)
87              break;
88      }
89      return 0;
90  }
91
92  void create(struct node **hptr)
93  {
94      struct node *newnode,*temp;
95       int item;
```

```
94      struct node *newnode,*temp;
95       int item;
96       newnode =(struct node *) malloc (sizeof(struct node));
97       printf("Enter the data : ");
98       scanf("%d",&item);
99       newnode->data=item;
100      if (*hptr==NULL)
101       {
102         newnode->next=NULL;
103        *hptr=newnode;
104       }
105      else
106     {
107     temp=*hptr;
108        while(temp->next!=NULL)
109        {
110                  temp=temp->next;
111         }
112       temp->next=newnode;
113       newnode->next=NULL;
114
115   }
116 }
117
118 void display(struct node *hptr)
119 {
120     struct node *ptr=NULL;
121     ptr=hptr;
122
123     if(ptr==NULL)
124     {
125        printf("Nothing to print\n");
```

```
123         if(ptr==NULL)
124         {
125             printf("Nothing to print\n");
126         }
127         else
128         {
129             while(ptr!=NULL)
130             {
131             printf("%d ",ptr->data);
132             ptr=ptr->next;
133             }
134         }
135     }
136
137     void sort(struct node *hptr)
138     {
139         if(hptr == NULL)
140             printf("Empty list\n");
141         else
142         {
143             int swap;
144             struct node *first = NULL;
145             struct node *last = NULL;
146             do
147             {
148                 swap = 0;
149                 first = hptr;
150
151                 while(first->next != last)
152                 {
153                     if(first->data > first->next->data)
154                     {
```

main.c

```
155                     int temp = first->data;
156                     first->data = first->next->data;
157                     first->next->data = temp;
158                     swap = 1;
159                 }
160                 first = first->next;
161             }
162             last = first;
163         }while(swap);
164     }
165 }
166
167 void reverse(struct node **hptr)
168 {
169     if(*hptr == NULL)
170     {
171         printf("Empty list\n");
172     }
173     else
174     {
175         struct node *prev, *curr, *head=*hptr;
176         prev = head;
177         curr = head->next;
178         head = head->next;
179
180         prev->next = NULL;
181
182         while(head!=NULL)
183         {
184             head = head->next;
185             curr->next = prev;
186
187             prev = curr;
```

main.c

```
181
182             while(head!=NULL)
183             {
184                 head = head->next;
185                 curr->next = prev;
186
187                 prev = curr;
188                 curr = head;
189             }
190
191             *hptr = prev;
192         }
193     }
194
195     void concatenate(struct node *hptr1, struct node *hptr2)
196     {
197         if(hptr1 == NULL && hptr2 == NULL)
198             printf("Both are empty lists\n");
199         else if(hptr1 == NULL || hptr2 == NULL)
200             printf("One of them is empty\n");
201         else
202         {
203             struct node *temp1 = hptr1;
204             struct node *temp2 = hptr2;
205             while(temp1->next != NULL)
206                 temp1 = temp1->next;
207             temp1->next = temp2;
208         }
209     }
```