link list application.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");
        printf("Enter your choice:");
        scanf("%d",&choice1);

        if(choice1 == 1)
        {
            printf("List1\n");
            while(choice!=5)
            {
            printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display \n5. Quit\n");
            printf("Enter your choice : ");
            scanf("%d",&choice);
```

```
        printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display \n5. Quit\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);
        if(choice == 1)
        {
            create(&head1);
        }
        else if(choice == 2)
        {
            sort(head1);
        }
        else if(choice == 3)
        {
            reverse(&head1);
        }
        else if(choice == 4)
            display(head1);
        else if(choice == 5)
            break;
        }
    }
    else if(choice1 == 2)
    {
        int choice2;
        printf("List 2\n");
      while(choice2!=5)
        {
        printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display \n5. Quit");
        printf("Enter your choice : ");
        scanf("%d",&choice2);
        if(choice2 == 1)
        {
            create(&head2);
        }
        else if(choice2 == 2)
        {
```

link list application.c X

```
                create(&head2);
            }
            else if(choice2 == 2)
            {
                sort(head2);
            }
            else if(choice2 == 3)
            {
                reverse(&head2);
            }
            else if(choice2 == 4)
                display(head2);
            else if(choice2 == 5)
                break;
            }
        }
        else if(choice1 == 3)
            concatenate(head1, head2);
        else if(choice1 == 4)
            display(head1);
        else if(choice1 == 5)
            break;
    }
    return 0;
}

void create(struct node **hptr)
{
    struct node *newnode,*temp;
    int item;
    newnode =(struct node *) malloc (sizeof(struct node));
    printf("Enter the data : ");
    scanf("%d",&item);
    newnode->data=item;
    if (*hptr==NULL)
    {
```

link list application.c

```
    if (*hptr==NULL)
     {
       newnode->next=NULL;
      *hptr=newnode;
     }
     else
    {
    temp=*hptr;
       while(temp->next!=NULL)
        {
                  temp=temp->next;
         }
       temp->next=newnode;
       newnode->next=NULL;

   }
}

void display(struct node *hptr)
{
    struct node *ptr=NULL;
    ptr=hptr;

    if(ptr==NULL)
    {
        printf("Nothing to print\n");
    }
    else
    {
        while(ptr!=NULL)
       {
        printf("%d ",ptr->data);
        ptr=ptr->next;
       }
    }
}
```

link list application.c

```
}

void sort(struct node *hptr)
{
    if(hptr == NULL)
        printf("Empty list\n");
    else
    {
        int swap;
        struct node *first = NULL;
        struct node *last = NULL;
        do
        {
            swap = 0;
            first = hptr;

            while(first->next != last)
            {
                if(first->data > first->next->data)
                {
                    int temp = first->data;
                    first->data = first->next->data;
                    first->next->data = temp;
                    swap = 1;
                }
                first = first->next;
            }
            last = first;
        }while(swap);
    }
}

void reverse(struct node **hptr)
{
    if(*hptr == NULL)
    {
```

link list application.c

```
void reverse(struct node **hptr)
{
    if(*hptr == NULL)
    {
        printf("Empty list\n");
    }
    else
    {
        struct node *prev, *curr, *head=*hptr;
        prev = head;
        curr = head->next;
        head = head->next;

        prev->next = NULL;

        while(head!=NULL)
        {
            head = head->next;
            curr->next = prev;

            prev = curr;
            curr = head;
        }

        *hptr = prev;
    }
}

void concatenate(struct node *hptr1, struct node *hptr2)
{
    if(hptr1 == NULL && hptr2 == NULL)
        printf("Both are empty lists\n");
    else if(hptr1 == NULL || hptr2 == NULL)
        printf("One of them is empty\n");
    else
```

link list application.c X

```
        struct node *prev, *curr, *head=*hptr;
        prev = head;
        curr = head->next;
        head = head->next;

        prev->next = NULL;

        while(head!=NULL)
        {
            head = head->next;
            curr->next = prev;

            prev = curr;
            curr = head;
        }

        *hptr = prev;
    }
}

void concatenate(struct node *hptr1, struct node *hptr2)
{
    if(hptr1 == NULL && hptr2 == NULL)
        printf("Both are empty lists\n");
    else if(hptr1 == NULL || hptr2 == NULL)
        printf("One of them is empty\n");
    else
    {
        struct node *temp1 = hptr1;
        struct node *temp2 = hptr2;
        while(temp1->next != NULL)
            temp1 = temp1->next;
        temp1->next = temp2;
    }
}
```