```
#include <stdio.h>
#include <stdlib.h>
struct node {
        int data;
        struct node*next;
};
void insertAtEnd (struct node * *head, int d) {
    struct node *temp,*n;
    if(*head==NULL){
            temp=(struct node*)malloc(sizeof(struct node));
            temp->data=d;
            temp->next=NULL;
            *head=temp;
    }
    else{
        temp=*head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        n=(struct node*)malloc(sizeof(struct node));
        n->data=d;
        n->next=NULL;
        temp->next=n;
    }
}
void reverse(struct node **head){
    struct node *prev,*cur,*next1;
    cur=*head;
    prev=NULL;
    next1=NULL;
```

```
if (*head == NULL) {
        printf("Empty List\n");
        return;
}
while(cur != NULL){
next = cur->next;
cur->next = prev;
prev = cur;
cur = next;
}
*head = prev;
}
void concat (struct node **head, struct node **head2) {
    if (*head == NULL) {
        *head = *head2;
        return;
    }
    if (*head2 == NULL) {
        *head2 = *head;
        return;
    }
    struct node* temp = *head;
    while(temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = *head2;
}
```

```
struct node * merge (struct node *a, struct node *b) {
    if (a==NULL){
        return b;
    }
    if (b== NULL) {
        return a;
    }
    struct node *c = NULL;
    if (a->data <= b->data) {
        c = a;
        c->next = merge (a->next, b);
    }
    else {
        c = b;
        c->next = merge (a, b->next);
    }
    return c;
}

struct node *midpoint (struct node* head ) {
    if (head == NULL || head -> next == NULL) {
        return head;
    }
    struct node*fast = head -> next;
    struct node*slow = head;
    while (fast != NULL && fast -> next != NULL){
        fast = fast->next->next;
        slow= slow -> next;
    }
    return slow;
}
```

```
Struct node * mergesort ( struct node * head ) {
    if( ( head == NULL ) || head->next == NULL ) {
      return head;
    }

  Struct node * mid = midpoint ( head );
  struct node * a = head;
  struct node * b = mid -> next;
  mid -> next = NULL;
  a = mergesort (a);
  b = mergesort (b);
  struct node * c = merge (a, b);
  return c;
}
void display ( struct node * head ) {
      while ( head ) = NULL ) {
            printf ("%d ->", head -> data);
            head = head -> next;
      }
      printf ("\n");
  }
  int main()
  {
  struct node * head 1 = NULL, * head 2 = NULL, * head 3 = NULL,
              * head 4 = NULL, * ans = NULL;

  int data, n;
  Printf ("... sorting .... \n");
  Printf ("enter the list to be sorted ( enter -1 to stop ): \n");
  scanf ("%d", &data);
  while ( data != -1 ) {
```

```
insertAtEnd (&head1, data);
scanf("%d", &data);
}
printf(" list before sorting :");
display (head1);
ans = mergesort (head1);
printf(" list after sorting:");
display (ans);
printf("\n.... Reverse--- \n");
printf(" enter the list to be reversed ( Enter -1 to stop): \n");
scanf("%d", &data);
while (data != -1) {
        insertAtEnd (&head2, data);
        scanf("%d", &data);
}
printf(" list before reversing:");
display( head2);
reverse (head2);
printf(" list after reversing:");
display(head2);
printf("\n--- concatenation -- \n");
printf(" Enter the first list ( Enter -1 to stop): \n");
scanf("%d", &data);
while( data != -1) {
        insertAtEnd (&head3, data);
        scanf("%d", &data);
}
printf(" Enter the second list ( Enter -1 to stop): \n");
scanf("%d", &data);
```

```
while (data != -1) {
        insertAtEnd (&head4, data);
        scanf ("%d", &data);
}
  printf (" First list: ");
  display (head3);
  printf (" second list: ");
  display (head4);
 concat (&head3, &head4);
 printf (" concatated list: ");
 display (head3);
 return 0;
}
```