1. Implementation of sorting, reverse & link 2 Linked List

```
#include <stdio.h>
#include <stdlib.h>
Struct node {
    int value;
    Struct node * next;
};
type def struct node * node;
NODE getnode (){
    NODE new node = (NODE) malloc (seize of (Struct node));
    if (new_node == NULL){
    printf ("memory allocation failed \n");
    exit (1);
    }
    Return new node;
}
NODE insert_end (int item, NODE first){
    NODE new_end = getnode ();
    new_end -> value = item;
    new_end -> next = NULL;
if (first == NULL ) {
    return new_end;
}
NODE current = first;
while (current -> next != NULL ){
    current = current -> next;
}
current -> next = new_End;
return first;
}
```

```
NODE reverse (NODE first) {
NODE current, first, temp;
   current = NULL;
while (first != NULL) {
   temp = first;
   first = first -> next;
   temp -> next = current;
   current = temp;
   }
   return current;
}
NODE concatenate (NODE first_1, NODE first_2) {
if (first_1 == NULL) return first_2;
if (first_2 == NULL) return first_1;
NODE last1 = first_1;
while (last1 -> next != NULL) {
   last1 = last1 -> next;
}
last1 -> next = first_2;
return first_1;
}
NODE sort (NODE first) {
   if (first == NULL || first -> next == NULL) {
   return first;
   }
NODE temp1, temp2;
int temp_value;
for (temp1 = first; temp1 != NULL; temp1 = temp1 -> next) {
   for (temp2 = temp1 -> next; temp2 != NULL; temp2 =
      if (temp1 -> value > temp2 -> value) {
         temp_value = temp1 -> value;
         temp1 -> value = temp2 -> value;
         temp2 -> value = temp_value;
} } }
```

```
return first;
}
Void display(NODE first){
  if (first == NULL){
     printf("Linked list is empty\n");
     return;
    }
   NODE temp= first ;
  while (temp != NULL){
     printf("%d\t", temp->value);
     temp = temp -> next ;
  }
   printf("\n");
}
int main(){
   NODE first_1 = NULL;
   NODE first_2 = NULL;
    int choice , item;
while (1){
   printf("\n Menu:\n");
   printf("1. ins || 1 ");
   printf("2. ins || 2 ");
   printf("3 Sort || 1 ");
   printf("4 sort || 2");
   printf("5 rev || 1 ");
   printf("6 rev || 2 ");
   printf("7 concat || ") ;
   printf("8 disp || 1 ");
   printf("9 disp || 2 ");
   printf("10. exit\n");
   printf("enter you choice ");
   scanf("%d", &choice);
```

```
switch (choice) {
case 1:
    printf ("enter value to insert");
    scanf ("%d", &item);
    first_1 = insert_end (item, first_1);
    break;
case 2:
    printf ("enter value to insert");
    scanf ("%d", &item);
    first_2 = insert_end (item, first_2);
    break;
case 3:
    printf ("sorting LL1 \n");
    first_1 = sort (first_1);
    break;
case 4:
    printf ("sorting LL2 \n");
    first_2 = sort (first_2)
    break;
case 5:
    printf ("LL1 being reversed");
    first_1 = reverse (first_1);
    break;
case 6:
    printf ("LL2 being reversed");
    first_2 = reverse (first_2);
    break;

case 7:
    first_1 = cancate (first_1, first_2);
    break;
case 8:
    display (first_1);
    break;
```

```
Case 9:
   display (first_2);
    break;
Case 10:
   exit (0);
default:
   printf ("invalid choice.\n");
}
}
return 0;
}
```

O/P

```
Menu:
1 insert 1
2 insert 2
3 Sort 1
4 sort 2
5 reverse 1
6 reverse 2
7 concate 2 1
8 display 1
9 display 2.
Enter you choice:
1
11
1
22
1
33
```

8
33 22 11

2
44
2
66
2
55
3
11 sorted
9
66 55 44

7
3
11 22 33 66 55 44

26/11/24

```
#include <stdio.h>
#include <stdlib.h>
struct NODE {
 int data;
 struct Node * next;
 };
void display (struct NODE *head ) {
 if (head == NULL) return;
 struct Node * temp = head ;
 do{
   printf ("%d ", temp -> data);
    temp = temp -> next;
   }
  while (temp != head);
  printf ("\n");
}
struct Node * insert ( struct Node *head , int val, int pos ){
    struct Node * newNode = (struct Node * ) malloc (sizeof (
                                    struct Node)) ;
       newNode -> data = val;
      if (head == NULL) {
       newNode -> next = newNode ;
       return NewNode;
       }
 struct Node * temp = head
if (pos == 0) {
   while (temp -> next != head) temp = temp -> next;
    temp -> next = newNode;
    newNode -> next = head;
    return newNode;
 }
 if (pos == -1) {
```

```
while (temp -> next != head ) temp = temp -> next;
 temp -> next = newNode;
 newNode -> next = head;
 return head;
for ( int i = 0; i < pos - 1; i++ ) temp = temp -> next;
 newNode -> next = temp -> next;
 temp -> next = newNode
 return head;
}
Struct Node * delete ( Struct Node *head, int pos ) {
 if (head == NULL ) return NULL;
 Struct Node * temp = head;
 if (pos == 0 ){
 while (temp -> next != head) temp = temp -> next;
 Struct Node * delNode = head;
 temp -> next = head -> next;
 head = head -> next;
 free (delNode);
 return head ; }
 if ( pos == -1 ) {
  while ( temp -> next != head ) temp = temp -> next;
  Struct Node * delNode = temp -> next;
  temp -> next = temp -> next -> next;
  free (delNode);
  return head;
 }
 for (int i = 0; i < pos - 1; i++) temp = temp ->
  next;
 Struct * delNode = temp -> next -> next;
  free (delNode);
  return head;
  }
```

```
int main() {
struct Node *head = NULL;
int choice, val, pos;
while (1) {
printf ("%d", &choice);
switch(choice){
case 1:
   printf (" enter value & position (use -1 for end and 0 for
     start): ");
   scanf ("%d %d", &val, &pos);
   head = insert (head, val, pos);
   break;
case 2:
   printf (" enter position to delete (use -1 for end & 0 for
         start:) ");
   scanf ("%d", &pos );
   head = delete (head, pos);
   break;
case 3:
   display (head);
   break;
Case 4:
   exit (0);
}
}
}
```

1. insert
2. delete
3. display
4. Exit.
Enter value and position (use -1 for end and 0 for start): 10
0
1 insert
2. delete
3. display
4. Exit.
1.
20
0
1.
30
0
enter choice 3:

30 20 10

1. insert
2. delete
3. display
4. Exit
1.
Enter value and position (use -1 for end and 0 for start) 40
-1
enter your choice 3
30 20 10 40.

enter you choice : 2
Enter your choice (1 from start and 0 from beggining) :
0.
3
30 20 10

enter your choice 2
0
3
20 10.
4.
Exited.

26/11/24