1. Single linked list - sort, reverse, concatination.

Sort a list.

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

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

struct Node * swap( struct Node * ptr1 , struct Node * ptr2)
{
 struct Node * tmp = ptr2 -> next ;
 ptr2 -> next = ptr1
 ptr1 -> next = tmp;
 return ptr2 ;
}
int bubblesort (struct Node ** head, int count)
{
 Struct Node **h;
int i, j , swapped ;

for (i=0 ; i<= count ; i++)
{
 h= head;
 swapped = 0 ;

for (j=0 ; j< count - i -1 ; j++)
{  struct Node * p1 = *h
   struct Node * p2 = p1 -> next ;
```

```
if (p1->data > p2->data)
{
    *h = swap(p1, p2);
    swapped = 1;
}
h = &(*h)->next;
}
if (!swapped == 0)
  break;
}
}
void printlist (struct Node* n)
{
  while(n != NULL) {
   printf("%d ->", n->data);
   n = n->next;
}
printf("\n");
}
void insertATbegin(struct Node** start_ref, int data)
{
 struct Node* ptr1
  = (struct Node*) malloc(sizeof(struct node));

ptr1->data = data;
ptr1->next = *start_ref;
*start_ref = ptr1;
```

```
int main()
{
 int arr[] = {78, 20, 10, 32, 1, 5};
 int list_size, i;

 struct Node *start = NULL;
 list_size = sizeof(arr) / sizeof(arr[0]);

 for (i = list_size - 1; i >= 0; i--)
     insertAtBegin(&start, arr[i]);
 printf("Linked list before sorting \n");
 printList(start);

 bubbleSort(&start, list_size);
 printf("Linked list after sorting \n");
 printList(start);

 return 0;
}
```

Output:

Linked list before sorting.
78 -> 20 -> 10 -> 32 -> 1 -> 5 ->

Linked list after sorting
1 -> 5 -> 10 -> 20 -> 32 -> 78 ->

Reverse a linked list:

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

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

static void reverse (struct Node ** head_ref)
{
   struct Node * prev = NULL;
   struct Node * current = * head_ref;
   struct Node * next = NULL;
   while (current != NULL) {
     next = current -> next;
     current -> next = prev;
     prev = current;
     current = next;
   }
   * head_ref = prev;
}

void push (struct Node ** head_ref, int new_data)
{
   struct Node * new_node
      = (struct Node *) malloc (sizeof (struct Node));
   new_node -> data = new_data;
   new_node -> next = (* head_ref);
   (* head_ref) = new_node;
}
```

```
void printlist (struct Node * head)
{
 struct Node * temp = head;
while (temp != NULL)
{ printf ("%d ", temp -> data);
  temp = temp -> next;
}
}
int main()
{
  struct Node * head = NULL;
  push (& head, 20);
push (& head, 4);
Push (& head, 15);
Push (& head, 85);

Printf ("Given linked list \n");
printlist (head);
Reverse (&head);
printf ("In Reversed linked list \n");
Printlist (head);
getchar();
}
```

output:

Given list list
85 15 4 20

Reversed linked list
20 4 15 85

* Concatination of two linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node * next;
} *temp=NULL, *first=NULL, *second=NULL;

struct Node * Create (int A[], int n)
{
 int i;
struct Node *t, *last;
temp = (struct Node *) malloc (size of (struct Node));
temp -> data = A[0];
temp -> next = NULL;
last = temp;

for(i=1; i<n; i++)
{
 t = (struct Node *) malloc (size of (struct Node));
t -> data = A[i];
t -> next = NULL;
last -> next = t;
last = t;
}
 return temp;
}
```

```
void display (struct Node *p)
{
  while (p != NULL)
  {
    printf ("%d", p -> data);
    p = p -> next;
  }
}

void concat (struct Node * first, struct Node * second)
{
  struct Node *p = first;
  while (p -> next != NULL)
  {
    p = p -> next;
  }
  p -> next = second;
  second = NULL;
}

int main ()
{
  int A[] = { 9, 7, 4, 3 };
  int B = []= { 2, 5, 6, 8 };

  first = Create (A, 4);
  second = Create (B, 4);

printf ("\n 1st Linked List : ")
Display (first);
```

printf ("\n 2nd linked list :");
Display (second);

Concat (first, second);

printf ("\n\n Concatenated linked list : \n");
Display (first);
return 0;
}

Output:

1st linked list : 9743
2nd linked list : 2568

Concatenated linked list :

9743256 8