

1. Single Linked List - sort, reverse, concatenation.

sort in 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;
```

Done
Page

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

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

    struct Node *start = NULL;
    list_size = size_of(arr) / size_of(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 ("\n Reversed linked list \n");
printlist (head);
getchar();
```

Output:

Given list list
85 15 4 20

Reversed linked list
20 4 15 85

+ Implementation 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 (sizeof (struct Node));

temp -> data = A[0];

temp -> next = NULL;

last = temp;

for(i=1; i < n; i++)

{

t = (struct Node *) malloc (sizeof (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;
    }
}
```

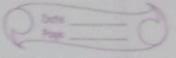
```
void concat (struct Node *first, struct Node *second)
{
    struct Node *p = first;
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = second;
    second->next = NULL;
}
```

```
int main ()
```

```
int A [] = { 9, 7, 4, 34 };
int B [] = { 2, 5, 6, 84 };
```

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

```
printf ("In 4th Linked List : ")
Display (first);
```



printf ("[n]nd linked list :");
Display (second);

concat (first, second);

printf ("[n]n concatenated linked list :\\n");
Display (first);
return 0;

4

Output:

1st linked list : 9743

2nd linked list : 8567

Concatenated linked list :

97432567

✓
97
85
2567