**Circular Linked List**

```
#include<stdlib.h>

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

void insert(struct node *h, int data){
    struct node *temp = (struct node *) malloc(sizeof(struct
node));
    temp->data = data;
    if(h->next == NULL){
        temp->next = h;
        h->next = temp;
    }
    else {
        temp->next = h->next;
        h->next = temp;
    }
}

int length(struct node *h){
    struct node *ptr = h->next;
    int len =0;
    while (ptr != h)
    {
        len++;
        ptr = ptr->next;
    }
    return len;
}

void add(struct node *h, int data){
    struct node *temp = (struct node *) malloc(sizeof(struct
node));
    temp->data = data;
    if(h->next == NULL){
        temp->next = h;
        h->next = temp;
        return ;
    }
     struct node *ptr;
    ptr = h->next;
    for(int i =0; i<length(h)-1; i++){
        ptr = ptr->next;
    }
    temp->next = ptr->next;
    ptr->next = temp;
}

void display(struct node *h){
```

```
    struct node *ptr = h->next;
    while (ptr != h)
    {
        printf("%d ",ptr->data);
        ptr = ptr->next;
    }
    printf("\n");
}

void delete(struct node *h){
    struct node *temp = h->next;
    h->next = h->next->next;
    free(temp);
}
void deletePos(struct node *header, int key)
{

     struct node *ptr = header->next,*temp;
     temp = header->next;

     if(temp->data == key) {
          header->next = header->next->next;
          return;
     }
     while(ptr!=NULL){
          if(ptr->data == key){
               temp->next=ptr->next;
               free(ptr);
            return;
          }
          else{
               temp = ptr;
               ptr = ptr->next;

          }
     }
}
void sortL(struct node *head){
     int i=0, j=0, len = length(head);
     struct node *temp1=head->next, *temp = head->next;
     for(i=0; i<len; i++){
          for(j=0; j<len; j++){
               if(temp1->data < temp->data){
                    int T = temp1->data;
                    temp1->data = temp->data;
                    temp->data = T;
               }
               temp = temp->next;
          }
          temp1 = temp1->next;
          temp = head->next;
     }
```

```
}

void reverseList(struct node *h) {
     struct node *ptr, *after, *prev, *last;
     prev=h->next;
     last=prev;
     ptr=prev->next;
     while(ptr->next!=h)
     {
          after=ptr->next;
          ptr->next=prev;
          prev=ptr;
          ptr=after;
     }
     ptr->next=prev;
     h->next=ptr;
     last->next=h;
}

int main(){
    struct node *head;
    head = (struct node *) malloc(sizeof(struct node));

    head->next = NULL;
    add(head, 32);
    add(head, 43);
    add(head, 56);
    add(head, 98);
    add(head, 1);
    display(head);
    deletePos(head,32);
    display(head);
    sortL(head);
    display(head);
    reverseList(head);
    display(head);
    return 0;
}
```

**Output:**

**32 43 56 98 1 // after Insert**
**43 56 98 1 // after delete**
**1 43 56 98 // after sortL**
**98 56 43 1 // after reverseList**